AFIT/GE/ENG/99M-08

TRADE-OFF ANALYSIS OF
COMMUNICATIONS CAPABILITIES
OF INTER-SATELLITE LINKS

THESIS

Andrew J. Feltman, Captain, USAF

AFIT/GE/ENG/99M-08

19990413 077

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GE/ENG/99M-08

TRADE-OFF ANALYSIS OF COMMUNICATIONS CAPABILITIES OF

INTER-SATELLITE LINKS

THESIS

Presented to the Faculty of the Graduate School of Engineering

Of the Air Force Institute of Technology

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Andrew J. Feltman, B.S.
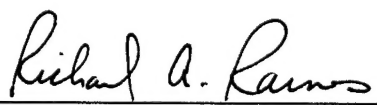
Captain, USAF

March 1999

AFIT/GE/ENG/99M-08

TRADE-OFF ANALYSIS OF COMMUNICATIONS CAPABILITIES OF

INTER-SATELLITE LINKS

Andrew J. Feltman, B.S.

Captain, USAF

Approved:

_____

RICHARD A. RAINES, Ph.D., Maj, USAF
Committee Chairman

3 Mar 99
date

_____

MICHAEL A. TEMPLE, Ph.D., Maj, USAF
Committee Member

3 Mar 99
date

Acknowledgements

Table of Contents

List of Figures

List of Tables

AFIT/GE/ENG/99M-08

Abstract

This thesis designs, develops, and uses software Graphical User Interfaces (GUIs) to analyze the communications capabilities of Radio Frequency (RF) Inter-Satellite Links (ISLs). The GUIs are geared towards analyzing the proposed ISLs of the Discoverer II program, but are general enough to permit analysis of any free-space RF ISLs. Discoverer II is a demonstration program of low-earth orbiting satellites and is primarily focused on satellite-based sensor technology. This thesis shows RF ISLs can meet the program requirement to broadcast the sensor data back to CONUS in near-real-time. The GUIs operate in real-time and explore trade-offs in the communications capability by varying the requirements, parameters, and components of the system design. Three GUIs are developed. The first GUI uses parameters from an assumed satellite constellation geometry to calculate the operating range of the ISLs. The second GUI solves for an unknown parameter of the communications system when all other parameters are given. The third GUI extends the analysis capability by plotting the trade-off between two parameters over a specified range of data. The analysis of the Discoverer II ISLs indicates that antenna radius is a very significant factor in the trade-off analysis, as well as being an important satellite design parameter.

Trade-Off Analysis of Communications Capabilities of Inter-Satellite Links

1 Introduction

1.1 Background/Motivation

The Discoverer II program is a demonstration program primarily focused on showing the technical feasibility of using satellite-based sensor technology. The primary sensor functions for the Discoverer II satellites are synthetic aperture radar imaging and ground moving target indication. The Discoverer II program is designed to supplement or replace the data available from the Joint Surveillance, Tracking, and Attack Radar System (JSTARS) aircraft. Space-based surveillance has three main advantages over aircraft-based surveillance. First, space-based surveillance does not place the aircraft or aircrews in harm's way and human factors (e.g., crew rest) and aircraft maintenance issues are avoided. Second, satellites can perform covert surveillance on an area prior to hostilities. This surveillance can be used to build terrain maps of an area and track enemy forces; JSTARS aircraft cannot perform this surveillance covertly. Third, satellites provide regional coverage under all conditions (day/night/all-weather), including areas deep within the enemy's control.

The Discoverer II program outlines several technical requirements for the system. The requirements of interest to this thesis follow. The Discoverer II satellite constellation will consist of 24 to 48 low-earth orbiting satellites (LEOsats), in circular orbits at an altitude of 770 km. The program specifies that the satellites will simultaneously

1

broadcast the sensor data down to tactical users in the area of responsibility and back to the continental United States in near-real-time. The system must be able to interface with existing ground assets using a proprietary data format which has a data rate of 512 Mbps. Inter-Satellite Links (ISLs) are a possible method to satisfy the requirement to send sensor data back to CONUS. This thesis investigates the possible use and capabilities of Inter-Satellite Links (ISLs) to transfer data between satellites [Whe98, Hug98].

1.2 Problem Statement

This thesis develops, implements, and validates software tools for analyzing the communications capabilities of Radio Frequency (RF) ISLs. Although the tools are geared towards analyzing the ISLs of the proposed satellites in the Discoverer II program, they are extremely flexible and can be used for analyzing any free-space RF ISL. The tools explore trade-offs in the communications capability by varying the requirements, parameters, and components of the communications system. The tools are intended for high-level analysis of the overall system design and performance.

1.3 Overview of Thesis

The first design consideration of an ISL is to determine whether the link will use an RF or laser communications (lasercom) system. This thesis examines both RF and lasercom systems. RF systems, and in particular, the RF link equation, are analyzed in detail. Current lasercom system technology is discussed, but the laser link equation is not examined in detail. Early background research indicated that lasercom ISLs would be more useful than RF ISLs. However, further research showed that RF ISLs were practical for the proposed ISLs of the Discoverer II program. This finding refocused

2

emphasis on RF systems and the RF link equation. Research into lasercom was not sufficient enough to permit development of software tools using the laser link equation.

As an introduction, this thesis examines orbital mechanics. The purpose of this is to calculate the ISL ranges. A graphical user interface (GUI) is designed to support the calculation of ISL range based on parameters and assumptions of the satellite constellation.

Two additional GUIs are built to analyze RF ISLs. The first solves for any unknown parameter in the RF link equation, given values for all other parameters. The second extends the analysis of the first by plotting trade-offs between two parameters over a range of data.

2 Background/Literature Review

2.1 Introduction

The design of any communications link involves trade-offs among the various parameters in the system. In designing an ISL, the first design trade-off that has to be made is determining whether the link will use an RF communications system or a laser communications system. Laser communications systems, while not nearly as mature as RF systems, have become viable candidates for ISLs in the past five to ten years because of rapid advances in laser system technologies. The analysis of both RF and laser communications systems is based upon the link equation, which details the sources of signal power, amplification, noises, and losses in the system. Different forms of the link equation are used when considering RF and laser systems because of physical differences in the communications systems. Because laser communications systems are less mature than RF systems, this thesis reviews state-of-the-art laser technologies that would most likely be considered in a laser communication system design. However, as a preliminary step, this thesis examines orbital mechanics. The purpose of this is to calculate the distance between satellites (i.e., the line of sight (LOS) range) for the ISL based upon various parameters of the satellite constellation. As it turns out, LOS range, and the associated free space loss, is the single most important factor in the link equation.

2.2 Orbital Range Calculations

Some knowledge of orbital mechanics is required to calculate the distance between satellites. This thesis uses three assumptions in making these calculations. The first (and main) assumption is that the satellites are in circular polar (90-degree inclination) orbits.

4

The 90-degree inclination assumption greatly simplifies the calculations, but two weaknesses of the assumption are analyzed at the end of this section. The second assumption is that the satellites are spaced evenly within their orbital planes. For instance, if there are six satellites in an orbital plane, the satellites are spaced every 60 degrees. The third assumption is that the orbital planes are evenly spaced around the earth. For instance, if there are four orbital planes, the planes will be spaced every 45 degrees. These two assumptions are reasonable for a completed constellation of satellites. The calculation of distance between adjacent orbital planes is based on the distance between the two planes at the equator. The distance between adjacent orbital planes in a constellation of polar-orbiting satellites is greatest at the equator and the inter-plane distance decreases with increasing latitude (both North and South) as the planes converge at the poles. While the satellites within each plane are equally spaced, the satellites in adjacent planes can be offset from each other, which increases the range between the satellites in adjacent orbital planes. The offset is typically specified in degrees and each plane is offset from its adjacent planes by the offset angle. Satellite constellations are often designed with offset geometry because it gives better coverage of the earth's surface compared to a geometry where adjacent orbital planes are not offset [AdR87].

Figure 1 shows how the distance between two satellites is computed. Because the satellites within an orbital plane are evenly spaced, the central angle (γ) from the center of the earth can be computed. Similarly, the central angle between adjacent orbital planes can be computed to give the distance between the planes at the equator. The angular offset between adjacent planes would have to be accounted for to determine the distance

5

between the actual satellites in adjacent orbital planes. The distance from the center of the earth to the satellite is the sum of the earth's mean radius (6378 km) and the altitude of the satellite (a). By the geometry of the isosceles triangle, the angles at each satellite (α) are equal. Applying the law of sines to compute the LOS range:

$$\text{LOS Range} = (r_e+a)\sin(\gamma)/\sin(\alpha) \tag{1}$$
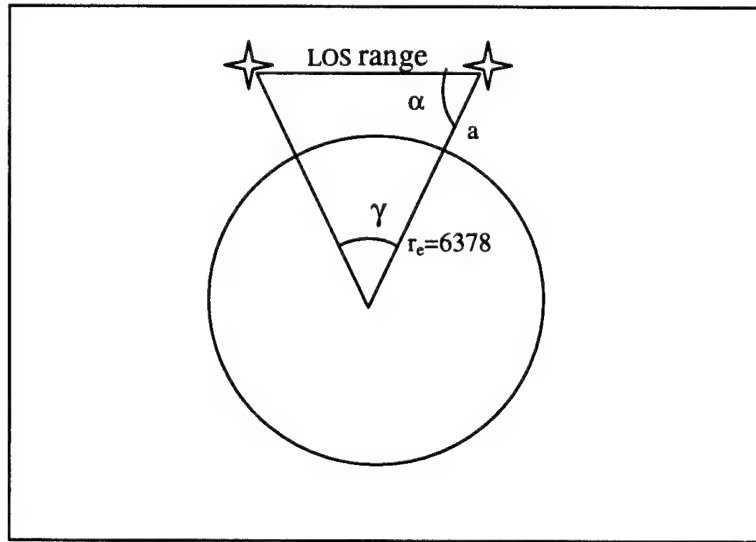
Figure 1. LOS Range Determination

An important consideration in the range calculation is the minimum LOS height above the earth's surface. If satellites at a given altitude are spaced (angle-wise) too far apart, the LOS crosses through a portion of the atmosphere. Communications that travel through the atmosphere experience fading and diffraction effects that are difficult to accurately model and are not accounted for in these calculations. In extreme cases, the angular separation causes the 'LOS' to graze the earth's surface, in which case 'LOS' is obviously blocked and the two satellites cannot communicate directly. A commonly used atmospheric height above which communications are considered to be free space is 120 km [MoO87, Kat87]. The minimum LOS height between the two satellites can be

computed as shown in Figure 2. The length of the normal leg can be computed by applying the law of sines; the minimum LOS height is determined by subtracting the earth's radius.

$$\text{Min LOS height} = (r_e + a)\sin(\alpha) - r_e \qquad (2)$$
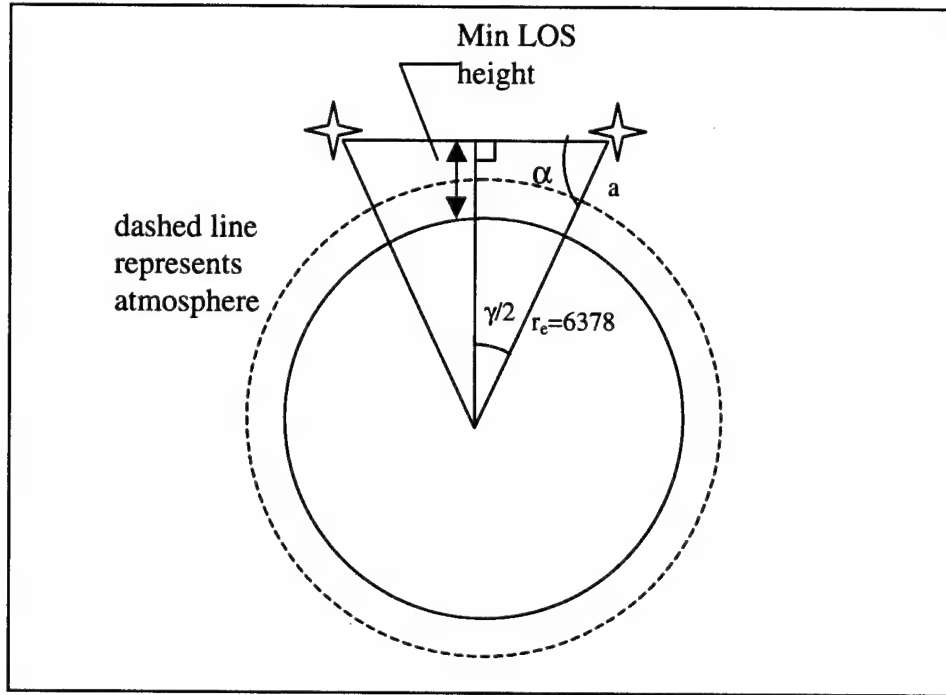
Figure 2. Minimum LOS Height

Figure 3.a. shows how the offset angle between adjacent orbital planes increases the distance between the actual satellites. The "LOS Range", computed in Equation 1, is the distance between the adjacent orbital planes at the equator. Given this and the offset angle (β, in degrees), the distance between the satellites (d') can be computed:

$$d' = (\text{LOS Range})/\cos(\beta) \qquad (3)$$

When the offset angle β is zero, the denominator of Equation 3 equals 1 and the distance between the satellites is the same as the distance between the adjacent planes at the equator. As β increases the denominator of Equation 3 decreases (less than 1) and the

7

distance between the satellites increases. Equation 3 approximates the adjacent orbital planes as parallel. In actuality, the orbital planes converge as they approach the poles; thus Equation 3 over-estimates the distance between the satellites. Figure 3.b. shows how to re-compute the minimum LOS height, which decreases as the distance between the
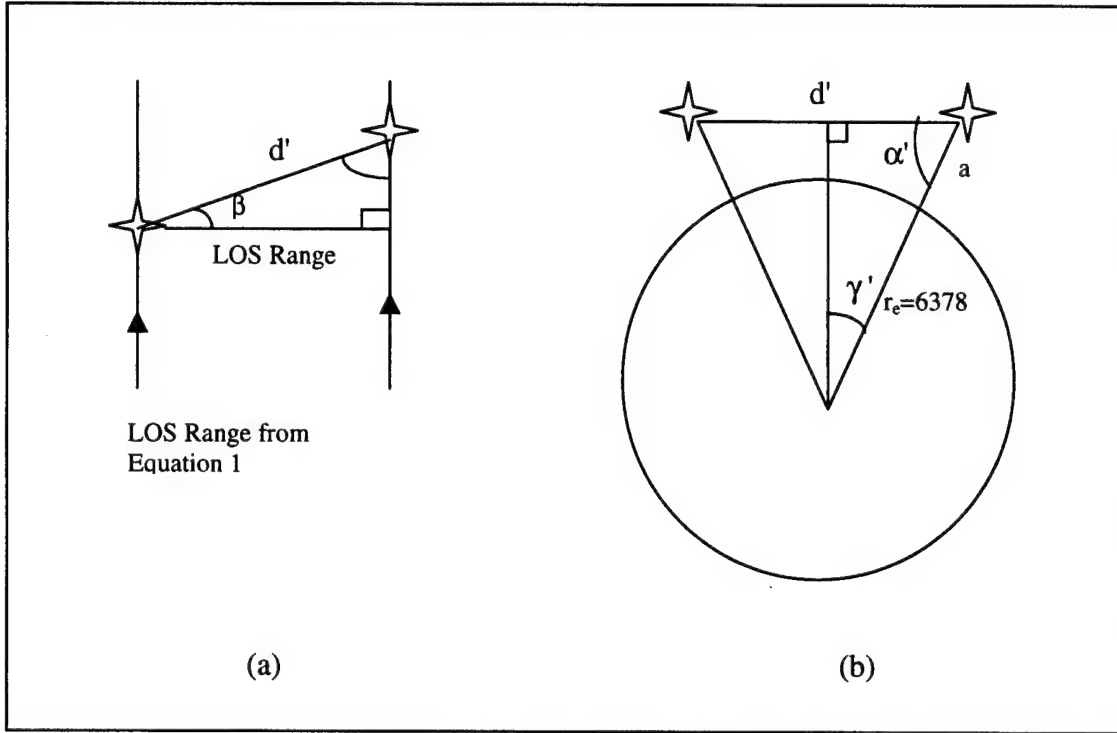
Figure 3. Angular Offset Geometry

satellites increases. Figure 3.b. is similar to Figure 2, except that angles γ' and α' (measured in radians) are initially unknown. These angles and the minimum LOS height can be computed as shown:

$$\gamma' = \sin^{-1}\left(\frac{d'/2}{r_e + a}\right) \tag{4}$$

$$\alpha' = \left(\pi/2 - \gamma'\right) \tag{5}$$

$$\text{Min LOS Height} = (r_e + a)\sin(\alpha') - r_e = (r_e + a)\sin(\pi/2 - \gamma') - r_e \tag{6}$$

8

The ranges determined by these calculations assume the satellites are in circular polar orbits. The calculations are not accurate if the orbits are non-circular. If the circular orbit assumption holds, the calculation of distance between satellites in the same orbital plane is accurate regardless of the plane's inclination. The distance between satellites in adjacent planes changes if the orbits are not polar. In practice, the polar orbit condition is very restrictive for two main reasons. First, if there is a constellation of satellites in polar orbits, there is a possibility that the satellites will collide at the poles where their orbital planes converge. This is one reason why, out of necessity, orbital planes are inclined. Second, satellite constellations with inclined orbital planes can achieve better coverage of the earth's surface compared to satellite constellations in polar orbits [AdR87].

2.3 Radio Frequency Communications Literature Review

RF communications systems are a mature technology. The concept of using RF ISLs has been around for more than 10 years. Motorola's Iridium system is the first commercial satellite system to use RF ISLs, showing that they are practical on a large network of Low Earth Orbiting (LEO) satellites. Figure 4 is a block diagram of the generic components of an ISL on a single satellite (each satellite has these components). In Figure 4, the solid arrows indicate the flow of the communications data; the dashed arrows indicate control signals.

The most basic consideration of a RF ISL is the carrier frequency. Two frequency bands are most commonly considered for use as RF ISLs. The first is the Ka-band at approximately 23.5 GHz; the Iridium system ISLs are in this band. The second is at approximately 60 GHz. These frequency bands correspond closely to peaks in atmospheric RF energy absorption. Water vapor causes an absorption peak at

9
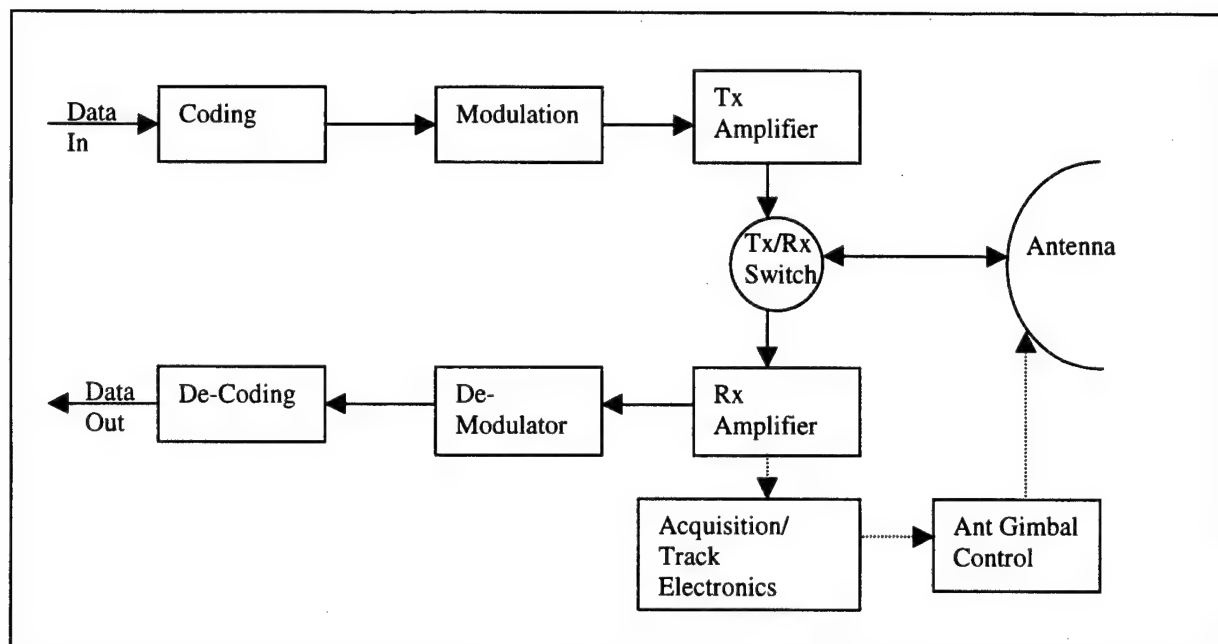
Figure 4. Generic ISL

approximately 22.2 GHz and oxygen causes a significantly stronger absorption peak at 60

GHz [Ric95]. Because of the atmospheric absorption in these frequency bands, these

frequency bands have limited practicality for use as uplink or downlink frequencies. As a

result of the atmospheric absorption, these frequency bands are practical for ISL use

because stray RF energy is likely to be attenuated and not cause interference with earth

station antennas (and vice-versa). The 60 GHz band was specifically designated for use

with ISLs by the World Administrative Radio Conference in 1979.

The primary tool for analyzing an RF ISL is the link equation. The link equation

allows the designer to trade-off various parameters and analyzes the system performance.

The link equation calculates sources of gains and losses in the transmitter, the channel,

and the receiver. Equation 7 shows a basic form of the link equation using decibel (dB)

units:

10

$$C/N_o = EIRP - L_T + [G_r - (kT)] \qquad (7)$$

where:

C/N$_o$ = The ratio of carrier energy to noise energy density at the receiver (dB-Hz)

EIRP = Effective Isotropic Radiated Power (dB)

L$_T$ = Total channel losses (dB)

G$_r$ = Receiver antenna gain (dB)

kT = Effective receiver noise energy density (dB-Hz).

The effective receiver noise energy density term contains two components; k is Boltzmann's constant = -228.6 (dB-Hz/K) and T is the effective system noise temperature (dB-K; to be discussed later). Also note that the link equation assumes the channel noise is modeled as Additive White Gaussian Noise (AWGN).

The transmitter's contribution to the link equation is EIRP. Equation 8 shows the components of EIRP (all units in dB):

$$EIRP = P_t - L_l + G_t \qquad (8)$$

where:

P$_t$ = Transmitter power

L$_l$ = Antenna line and coupling losses

G$_t$ = Transmit antenna gain.

Transmitter power is a major factor in determining the size, weight, power requirement, and cost of the communications system. Solid-state power amplifiers are probably the best choice for applications requiring approximately 20 W in the Ka-band. Traveling wave tube amplifiers are probably the best choice for higher-power applications in the Ka-band and applications in the 60 GHz band [Ric95]. The antenna line and coupling

losses depend on the actual length and construction (bends/joints) of the transmitter waveguide; a conventional value is 1 dB. The transmit antenna gain (on-axis, or boresight) is given by:

$$G_t = 10\log(4\pi A\eta/\lambda^2) \ \ (dB) \tag{9}$$

where:

A = Antenna physical aperture area

η = Antenna efficiency (typical values: $\eta = 0.55$ for circular apertures; $\eta = 0.7$ for horn apertures)

λ = Wavelength (meters).

The channel's contribution to the link equation is the channel loss term L_T. Equation 8 shows the components of L_T (all units in dB):

$$L_T = L_{FSL} + L_a \tag{10}$$

where:

L_{FSL} = Free Space Loss

L_a = Antenna pointing and polarization loss.

Free space loss is typically the largest single loss factor in the link equation.

$$L_{FSL} = 10\log(4\pi r/\lambda)^2 = 20\log(4\pi r/\lambda) \tag{11}$$

where:

r = Range between satellites (meters).

The polarization component of the antenna pointing and polarization loss can be considered negligible if circular polarization is used. The pointing component depends

on the half-power beamwidth of the antenna, the pointing accuracy of the satellite, and the antenna tracking accuracy (if the antenna is gimbaled). Equation 12 shows a general expression for the half-power beamwidth of an antenna:

$$BW_{3dB} = K\lambda/D \text{ (degrees)} \tag{12}$$

where:

K = Constant depending on the distribution of the aperture ($K = 70$ for tapered distributions; $K = 58$ for uniform distributions)

D = Aperture diameter (meters).

Equations 9 and 12 are valid for antennas with apertures, such as horn and dish antennas. Equations 9 and 12 do not hold for antennas that do not have apertures, such as dipole antennas.

Most antennas have tapered distributions (this is a worst-case assumption for beamwidth). Combining the pointing accuracy of the satellite and the antenna tracking accuracy into a single term, $\Delta\Theta$, Equation 13 shows an approximate expression for pointing loss (L_a units in dB) [Ric95, Joh93] :

$$L_a = 12((\Delta\Theta)D/70\lambda)^2 \tag{13}$$

Analyzing Equation 13, if the tracking accuracy equals the half-power beamwidth (Equation 12), the pointing loss will be 12 dB. In practice, satellites in geosynchronous orbits have tracking accuracy equal to one-tenth the half-power beamwidth, making pointing loss equal 0.12 dB. Because of the greater relative motion of LEOsats, their tracking accuracy is not as good as the tracking accuracy of geosynchronous satellites.

13

This thesis assumes the tracking accuracy of LEOsats is one-half the half-power beamwidth, making pointing loss equal 3 dB.

The receiver's contribution to the link equation is described by two terms: the receiver antenna gain and the effective receiver noise energy density. If the same antenna is used for transmitting and receiving (as shown in Figure 4), the receiver antenna gain is computed in the same manner as the transmit antenna gain (G_r units in dB):

$$G_r = 10\log(4\pi A\eta/\lambda^2) \tag{14}$$

Since antenna gain depends on frequency, the actual transmit gain equals the actual receive gain only if the same frequency is used for transmitting and receiving (which can be the case when one sense of circular polarization is used for transmitting and the other sense is used for receiving).

The effective receiver noise energy density is the product (in absolute units, not dB) of Boltzmann's constant and the effective system noise temperature. Expressed as dB and substituting in Boltzmann's constant:

$$\text{Effective receiver noise energy density (dB-Hz)} = -228.6 + T \tag{15}$$

The effective system noise temperature can be expressed as (T is in absolute units of K):

$$T = T_a/L + (L-1)T_o/L + T_r \tag{16}$$

where:

T_a = Antenna noise temperature; typically 10 K because the receive antenna is pointed at the transmitting satellite against a background of 'cold space'

L = Receiver line and coupling loss; as with the transmitter loss it is typically 1 dB

T_o = Ambient temperature; a conventional value is 300 K

T_r = Noise temperature of receiver front-end; a conventional value is 3000 K.

14

Substituting into Equation 16:

$$T = 3070 \text{ (K)} = 34.8 \text{ (dB-K)} \qquad (17)$$

A common figure of merit used to evaluate a receiver combines receiver gain and effective noise temperature. Using Equations 14, 16, and 17 (G_r/T units of dB-K^{-1}):

$$G_r/T = 10\log(4\pi A\eta/\lambda^2) \text{ (dB)} - T \text{ (dB-K)} = 10\log(4\pi A\eta/\lambda^2) - 34.8 \qquad (18)$$

Returning to the link equation (Equation 7) and substituting from Equations 8, 9, 10, 11, 14, 15, and 16:

$$C/N_o = P_t - L_l + 10\log(4\pi A\eta/\lambda^2) - 20\log(4\pi r/\lambda)$$

$$- L_a + 10\log(4\pi A\eta/\lambda^2) + 228.6 \qquad (19)$$

$$- 10\log(T_a/L + (L-1)T_o/L + T_r)$$

Assuming a circular antenna aperture (area $= A = \pi R^2$) and substituting values for L_l, L_a, and the temperature term (Equation 17), re-arranging the terms gives:

$$C/N_o = P_t - 1 + 40\log(2\pi) + 40\log(R) + 20\log(\eta) - 20\log(\lambda)$$

$$- 20\log(4\pi) - 20\log(r) - 3 + 228.6 - 34.8 \qquad (20)$$

Note that the $-20\log(\lambda)$ term is combined from the two gain terms and free space loss term. Simplifying all constants:

$$C/N_o = P_t + 40\log(R) + 20\log(\eta) - 20\log(\lambda) - 20\log(r) + 199.74 \text{ (dB)} \qquad (21)$$

Again, R is the antenna radius and r is the range. Equation 21 expresses the ratio of carrier energy to noise energy density available at the receiver (the Rx Amplifier block of Figure 4). This energy ratio is available at the receiver as a function of transmitter power.

15

Most digital communications systems specify a required probability of bit error of the received data (the data out, after demodulation and decoding). This probability of bit error, P_b, drives a requirement for a certain ratio of energy per bit to noise energy density, E_b/N_o. Equation 22 shows the relationship between required E_b/N_o and required C/N_o [Kat87]:

$$C/N_o = E_b/N_o + R_D + M \qquad (22)$$

where:

R_D = Information data rate (units of dB-bits/sec)

M = System link margin; a typical value is 2 dB for an ISL because free-space communications do not experience atmospheric fading that requires large link margins.

The requirement for a specified P_b drives a requirement for a specified E_b/N_o in conjunction with the demodulation and decoding used by the communication system. A typical value for P_b is 10^{-5} but higher values (10^{-2}) for voice data or lower values (10^{-12}) for critical data like satellite guidance commands may be required. The method of (de)modulation used by the system determines the relationship between P_b as a function of E_b/N_o [Skl88]. Coherently detected phase shift keying (PSK) is one of the most frequently used (de)modulation schemes. PSK schemes are commonly identified as "M-ary PSK," where M is the number of signals in the signal set and is typically a power of two. For 2-ary PSK (Binary PSK, or BPSK) and 4-ary PSK (Quadrature PSK, or QPSK), Equation 23 shows the relationship between P_b and E_b/N_o [Skl88]:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \qquad (23)$$

Note that E_b/N_o is often given in decibel form, but Equation 23 requires the absolute ratio. $Q(x)$, commonly called the Q-function, is defined as:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du \qquad (24)$$

The Q-function is the probability, or area, under the tail of a zero-mean, unit variance Gaussian distribution. It cannot be evaluated in closed form, but it can be approximated by simpler functions and it is commonly evaluated using tables. For coherently detected M-ary PSK, $M > 2$, an approximate relationship between P_b and E_b/N_o is given in Equation 25 [Skl88]:

$$P_b \approx \frac{2Q\left(\sqrt{\frac{2kE_b}{N_o}} \sin\frac{\pi}{M}\right)}{k} \qquad (25)$$

where $k = \log_2 M$ and the approximation improves as the energy-to-noise ratio increases.

Equation 25 assumes the M-ary PSK signals are Gray coded, which optimizes the probability of bit error. Notice that for $k = 2$, $M = 4$, Equation 25 simplifies to Equation 23. In Equations 23 and 25, solving for E_b/N_o given P_b requires 'reverse lookup' of the Q-function (i.e., finding the argument of the Q-function given the result). This is easily accomplished with tables. For coherently detected M-ary PSK schemes, as M increases, the required E_b/N_o to maintain a given P_b increases [PeZ95]. This behavior is shown in Figure 5.

17
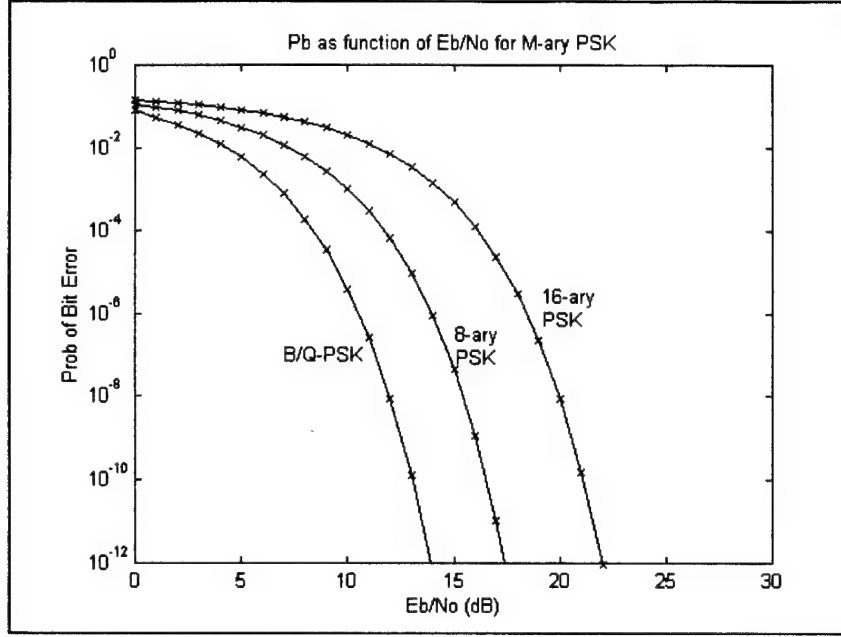
Figure 5. P_b versus E_b/N_o for M-ary PSK

The use of (de)coding essentially allows a system to use a reduced value of E_b/N_o while maintaining the required P_b. The amount of the reduction is called the *coding gain*. A typical coding scheme (length 7, rate 1/2, convolutional coding) provides a coding gain of 5.2 dB. As an example, an uncoded BPSK system with a $P_b = 10^{-5}$, requires $E_b/N_o = 9.6$ dB. Thus, a BPSK system with specified $P_b = 10^{-5}$ using this coding scheme requires:

$$E_b/N_o = 9.6 - 5.2 = 4.4 \text{ (dB)} \tag{26}$$

There are many different types of modulation and coding schemes that can be interchanged to provide multiple values of E_b/N_o that will meet the required P_b.

Equation 21 shows the amount of C/N_o available at the receiver as a result of the physical components of the RF system (i.e., the gains and losses in the transmitter, channel, and receiver). Equation 22 shows the amount of C/N_o required by the RF system to convert the received energy (the available C/N_o) into useful information for communication. The conversion occurs in the demodulator and decoding blocks of

18

Figure 4. If the available C/N_o (Equation 21) is less than the required C/N_o (Equation 22), the link will not be established. As a minimum, equality between Equations 21 and 22 is necessary:

$$E_b/N_o + R_D + M = P_t + 40\log(R) + 20\log(\eta) - 20\log(\lambda) - 20\log(r) + 199.74 \qquad (27)$$

If the available C/N_o is greater than the required C/N_o, the system is over-powered or wasting energy.

Equation 27 can be used to analyze trade-offs between various parameters of the system. To graph one parameter versus another, fix the other parameters and solve Equation 27 for one of the parameters of interest as a function of the other parameter of interest. A commonly graphed trade-off is antenna radius as a function of data rate.

Chapter 3 will discuss how the Matlab® Graphical User Interface (GUI) allows the parameters of the link equation to be set and solves the link equation.

2.4 Laser Literature Review

Laser communication (lasercom) systems are not as mature as RF communications systems. High-speed (hundreds of megabits per second) lasercom systems were first developed for use in guided media (fiber optics). Only recently have free space high-speed lasercom systems been developed [Kim98]. Lasercom systems have three primary advantages over RF communications systems, all of which are due to the fact laser frequencies are several orders of magnitude higher than RF. First, lasercom systems have higher information bandwidth than RF systems (the carrier frequency of a communications system limits the rate at which the system can transmit information). Second, laser beams have smaller beam divergence angles than RF signals. Third, lasers require smaller antennas (telescopes) than RF systems. The smaller beam divergence

19

angle of lasers means that pointing, acquisition, and tracking is more critical in a lasercom system compared to an RF system.

A generic lasercom system is shown in Figure 6 where solid lines indicate optical signals and dashed lines indicate electrical signals. Unlike Figure 4, the Signal In is already assumed to be coded; similarly, the Signal Out is still coded. The laser acts like the transmit amplifier of Figure 4. Laser modulation is usually different than RF modulation. The optics are a lasercom system's antenna. However, there is no need for a Tx/Rx switch in a lasercom system.
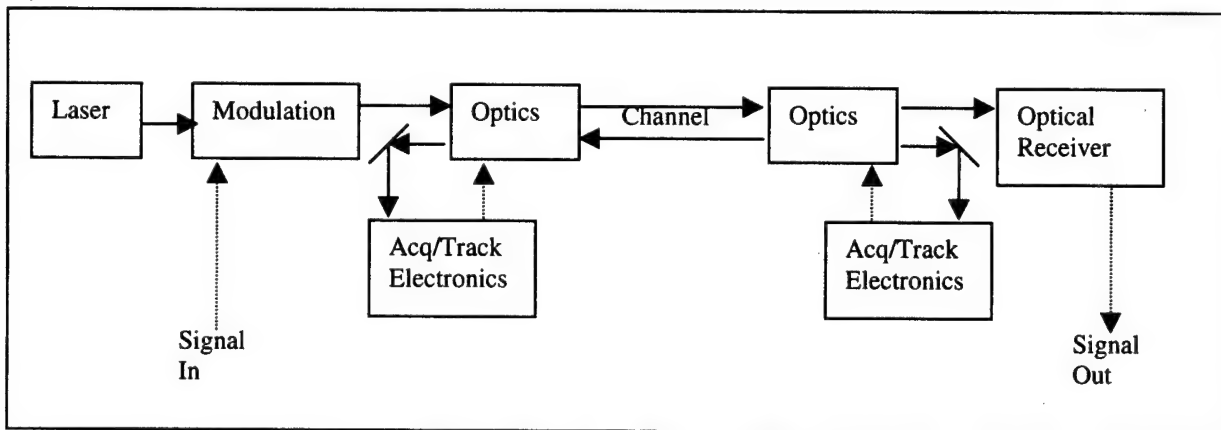
Figure 6. Generic Lasercom System

2.4.1 Laser Theory

Laser theory, in the most basic terms, is relatively simple. Lasing occurs when a resonant cavity discharges optical energy. Electrical or optical energy is 'pumped' into the cavity, the energy resonates and builds up in the cavity, and at a certain energy state or under certain conditions, the energy discharges. The primary types of lasers are discussed in the following sections. Lasers can operate in two modes: Continuous Wave (CW) or pulsed. CW lasers are lasing for long periods of time (seconds). Pulsed lasers emit their energy in very short pulses (on the order of nanoseconds). Pulsed lasers are

20

used in lasercom systems. The narrow (in time) pulses contain high peak power relative to average power and narrow pulses allow the laser to transmit information at high data rates.

2.4.1.1 Gas Lasers

The red He-Ne laser is the most common example of a gas laser. In a gas laser, a cavity (usually a glass tube) is filled with gas, electrodes in the tube allow electrical energy to 'pump' the cavity, and mirrors at the ends of the tube create the resonance. One of the mirrors is only partially reflective. The laser beam exits through this mirror. The most practical gas laser for a lasercom system is a CO_2 laser, which operates at 10.6 μm. This wavelength is not visible; it is in the far-IR portion of the spectrum. Overall, gas lasers are not the best candidates for ISLs because the electrodes and seals degenerate, causing the laser to fail.

2.4.1.2 Solid-state Lasers

The classic solid state laser is the ruby laser, where a ruby rod is pumped by a flashtube. The most practical solid-state lasers for ISLs are the Nd:YAG (Neodynium: Yttrium/ Aluminum/Garnet) laser and the frequency-doubled Nd:YAG laser, operating at 1.06 μm and 0.532 μm, respectively. Nd:YAG lasers are pumped by the laser light output of semiconductor lasers. Solid-state lasers are not the best candidates for ISLs because their modulation process is complex and because they tend to shift wavelength with age.

2.4.1.3 Semiconductor Lasers

Semiconductor lasers are 'lasers on a chip'. A cavity is formed within the semiconductor structure. Polished facets of the chip are used as the mirrors, with a current applied

through the chip to pump the laser. The pump current applied to the semiconductor directly modulates (on/off) the laser output. In Figure 6, the laser and modulation blocks would trade places (i.e., the modulating signal drives the laser). The most practical semiconductor laser for ISLs is the AlGaAs (aluminum, gallium, arsenic) laser operating around 0.85 μm. The wavelength is tunable depending on the manufacture of the chip. This benefit is discussed in Section 2.4.4.

2.4.1.4   Fiber Optic Amplifiers

Fiber optic amplifiers are not lasers, but they can be used to amplify the optical power of a laser. A fiber optic amplifier is a length of fiber optic cable that has been doped with certain amounts and types of chemicals. The information laser light enters one end of the fiber optic amplifier cable. The fiber optic amplifier cable is exposed to lasers operating at a different wavelength than the information laser. These 'pump' lasers cause the fiber optic amplifier to increase the optical power of the information laser (without changing its wavelength). The most common type is the Erbium Doped Fiber Amplifier (EDFA). The 1480 nm pumped EDFA uses an AlGaAs semiconductor laser for the information laser and is pumped with 1480 nm laser light (which can be produced from a different type of semiconductor laser). The 1480 nm pumped EDFA has a maximum output power of approximately 890 mW, but it is only 4% efficient (in terms of pump laser power to information laser power). The 980 nm pumped EDFA uses a 1550 nm information laser and has a maximum output power of approximately 450 mW with 8% efficiency. Fiber optic amplifiers have the potential for use in a lasercom ISL by increasing the low output power of semiconductor lasers. However, fiber optic amplifiers require a long length of fiber optic cable (approximately 20 m) and the pumping process is relatively inefficient.

As of the current time, fiber optic amplifier technology has not been integrated into lasercom ISL designs (or any satellite-based lasercom design), but the next generation of lasercom systems may be able to use this rapidly advancing technology [Ara98].

2.4.2 Laser Detection

A typical RF communication system uses heterodyne detection (i.e., the received signal is multiplied with a locally generated signal at a different frequency to produce an intermediate frequency). The data signal is then extracted from the intermediate frequency. In contrast, laser detection is usually done by direct energy detection. The laser detector is sensitive to the magnitude of received laser light, but is not sensitive to the amplitude or phase of the received laser light. The high-energy photons of laser light allow this operation. Direct detection is usually limited (by the photon energy and detector physical properties) to wavelengths shorter than 1 μm. CO_2 and other gas lasers have longer wavelengths and can be detected using coherent detection. Coherent detection uses a locally generated laser to extract phase information of the received laser beam. Coherent detection is not currently considered viable for use in ISLs. Direct detection systems experience some additive noise (from background radiation), but the primary noise source is the signal itself. The energy in the photons varies in a statistical manner. Since a detector cannot know a priori the energy in the photons, the detection process has inherent noise due to the signal.

2.4.3 Laser Link Equation

The laser link equation is the primary tool used to analyze a lasercom ISL. In a manner similar to the RF link equation, the laser link equation calculates the sources of

gains and losses in the transmitter, the channel, and the receiver. The exact form of the laser link equation is slightly different than the RF link equation:

$$M = 10\log(P) - L_t + G_T - L_p - L_n - L - L_{LINK}$$

$$+ G_R - L_R - 10\log(QE) - L_{proc} - 10\log(S_{req}) \qquad (28)$$

where:

M = System link margin (dB); as with RF ISLs, a value of 2 dB is typical

P = Laser output power (W); this value depends on the type and number of transmitting lasers

L_t = Transmit telescope feed (optics) losses (dB); modern telescopes use multi-element coated optics which typically cause a 3 dB loss

G_T = Transmit telescope gain (dB)

L_p = Transmit beam pointing loss (dB)

L_n = Transmit beam wavefront efficiency (dB)

L = Free space loss (dB) = $10\log(4\pi r/\lambda)^2$ = $20\log(4\pi r/\lambda)$; as with the RF link equation, free space loss is the single largest loss factor in the laser link equation

L_{LINK} = Additional losses (e.g., polarization and attenuation (dB); polarization losses are negligible when circular polarization is used, attenuation losses are negligible in free space)

G_R = Receiver telescope gain (dB) = $20\log(\pi D/\lambda)$; where D is the telescope aperture diameter; this is not exactly the same as for a RF system because of optical properties

L_R = Receiver telescope feed (optics) losses (dB); this is usually 3 dB like the transmit loss, but the transmit and receive optical paths are often not the same

QE = Energy detection efficiency of receiver detector

24

L_{proc} = Processing losses of system after received signal is converted from optical to electrical

S_{req} = Minimum signal required for performance, based on link noise background and detector noise.

Certain properties of optics need to be stressed. Some telescope designs have obscurations in their structures that block some of the light exiting a transmitting telescope. This obscuration is very important; sometimes transmit power is redefined to be the optical power exiting the telescope aperture, as opposed to the power entering the telescope. The narrow beam divergence of lasers makes pointing, acquisition, and tracking a very critical part of a lasercom design. The use of wide-beam beacon lasers with narrow-beam communications lasers and modern tracking systems can cause pointing losses of 3 dB (similar to pointing losses for a RF system).

2.4.4 State-of-the-Art

Free-space lasercom technology is not as mature as RF technology. For this reason, it is useful to examine state-of-the-art lasercom technology before designing a lasercom system. A lasercom system designed with technology that is in the lab or commercially available can be built more quickly and less expensively than a lasercom system designed with possibly unrealistic technology requirements. Semiconductor lasers are the best choice for ISLs compared to gas and solid-state lasers. Gas lasers have degeneration problems and the modulation process for solid-state lasers is complex and they tend to shift wavelength with age. Aluminum, gallium, arsenic (AlGaAs) semiconductor lasers are the most practical semiconductor lasers for use in ISLs. These lasers typically have

output power of approximately 100 mW [Cha98]. Current lasercom designs employ multiple lasers (at the same wavelength) each driven with the same signal to effectively increase the output power of 'the laser'. This design requires that each laser has its own optics, but that is not cost-prohibitive [Bis98]. A benefit of this design is that the combined laser power will degrade gracefully if individual lasers fail. AlGaAs lasers can be wavelength-tuned in the region of 830-865 nm by controlling the manufacture of the chip. The best optical filters in these wavelengths are approximately 10 nm wide. Therefore, it is possible to have four-channel wavelength (or frequency) division multiplexing using AlGaAs lasers [Bai98]. In a 'balanced' lasercom system, two channels each would be used for transmit and receive. AlGaAs lasers are directly modulated with the fastest 'driver' modulators rates of 622Mbps [Cha98].

2.5 Summary

This chapter provided background on the major components of an ISL. Basic orbital mechanics were introduced to calculate the range between satellites, because the free space loss associated with range is the most important loss factor in an ISL. RF systems were discussed in detail by examining the components of the link equation. The link equation is used calculate the effects of gains and losses throughout the RF system, i.e., in the transmitter, channel, and receiver. The Graphical User Interfaces discussed in Chapter 3 allows the link equation parameters to be entered and perform trade-off analysis of the RF system. Lasercom theory and the laser link equation were introduced. Since lasercom technology is not as mature as RF systems, current laser technology was discussed.

3 *Methodology*

3.1   Introduction

This thesis uses Matlab® to perform the computations required to analyze the Inter-Satellite Links (ISLs). Matlab® version 5.2. is used on both Windows PCs and Unix platforms. No specialized toolbox commands are required. Graphical User Interfaces (GUIs) are built using Matlab's® Guide tools. Guide is a set of tools that are specially designed to make it easier to build GUIs. With Guide, the components of a GUI (e.g., the buttons, text window, plots) are laid out graphically. Guide simplifies the process of changing the properties (e.g., color, text labels) of the GUI components. Guide also specifies which modules of Matlab® code (callbacks) are executed when a component of the GUI is acted upon (e.g., when a button is clicked, text is entered, an item in a drop-down box is selected). The code contained in the callbacks is what makes the GUI 'intelligent' (the code performs calculations, computes results, displays plots, etc.).

Several separate GUIs are used for analyzing the ISLs. Each GUI covers a broad functional area of the ISL. The GUIs are linked together so the user can progress from one GUI to another and so that data can be passed between GUIs. Separating the GUIs makes the callback coding easier and decreases the amount of clutter on any one GUI. Individual GUIs are described in detail below. Appendix A lists the filenames used in the GUIs. All of the GUIs share the characteristic that they return errors if alphabetic characters (or no characters at all) are entered as an input when numeric input is expected. The error message is returned in the Matlab® Command Window. The error

27

does not close the GUI, and if corrected (numbers are entered), the GUI continues functioning.

3.2 Satellite Range Calculations GUI

The Satellite Range Calculations GUI calculates information about the satellite constellation based on assumptions and equations presented in Section 2.2. The GUI is shown in Figure 7. The GUI uses five parameters: the number of satellites per orbital
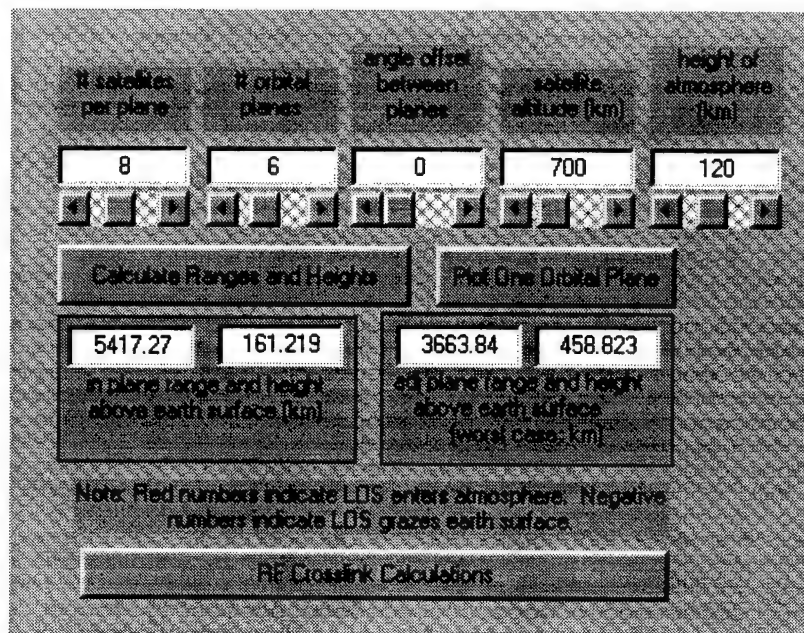
Figure 7. Satellite Range Calculations GUI

plane, the number of orbital planes, the offset angle between adjacent orbital planes, the satellite altitude, and the atmosphere height. The inputs can be directly entered in the edit boxes, or can be selected using the slider bars below each edit box. Any numeric value can be entered directly into the edit boxes; the slider bars have preset minimum and maximum values that are likely to be used when designing a constellation of low earth orbiting satellites. Table 1 lists the valid ranges and the preset slider bar ranges for the parameters. After the inputs are entered, clicking the "Calculate Ranges and Heights"

28

Table 1. Valid and Preset Ranges for Satellite Range Calculations GUI

Parameter	Valid Values		Slider Bar Presets	
	Min	Max	Min	Max
Number of Satellites per Orbital Plane	3	N/A	3	16
Number of Orbital Planes	2	N/A	2	16
Offset Angle	N/A[1]	N/A[1]	0	60
Satellite Altitude	N/A[2]	N/A	500	2000
Height of Atmosphere	N/A[2]	N/A	0	300

Notes:
[1]By convention, offset angle is in the range 0 to 90 degrees. For other values, the GUI will calculate physically invalid results, but will not return an error.
[2]Negative Altitude and Height are physically impossible. For negative values, the GUI will calculate physically invalid results, but will not return an error.

button will perform computations. Based on the number of satellites in each orbital plane and the satellite altitude, the In-Plane-Range and the (In-Plane Minimum) Height Above the Earth Surface are computed and displayed by the GUI. The In-Plane-Range is computed using Equation 1. The (In-Plane Minimum) Height Above the Earth Surface is equivalent to the minimum LOS height computed using Equation 2. Based on the number of orbital planes, the offset angle between adjacent orbital planes, and the satellite altitude, the Adjacent-Plane-Range and the (Adjacent-Plane Minimum) Height Above the Earth Surface are computed and displayed by the GUI. The Adjacent-Plane-Range is computed using Equations 1 and 3. The (Adjacent-Plane Minimum) Height Above the Earth Surface is computed using Equation 2. As shown in Figure 7, there are 8 satellites in each orbital plane, thus they are spaced 45 degrees apart. There are 6 orbital planes and the offset angle between adjacent orbital planes is zero. If the offset angle between adjacent orbital planes is changed to 22.5 degrees, the Adjacent-Plane-Range would increase from 3663.84 km to 3965.71 km and the (Adjacent-Plane

Minimum) Height Above the Earth Surface would decrease from 458.823 km to 416.583 km. The computed minimum height above the earth surface (for both in-plane and adjacent-plane) is compared to the given height of the atmosphere. If either minimum height is less than the height of the atmosphere, the computed minimum height is displayed with red text. If the LOS grazes the earth's surface (and would therefore be blocked), the computed minimum height is a negative number and will be displayed with red text. The red text is only a warning, it does not stop the GUI nor prevent the user from using the numbers in further calculation. Figure 8.a. shows satellite constellation parameters and computed ranges and heights such that the earth blocks the LOS. Figure 8.a. is similar to Figure 7, except the number of satellites per plane is reduced from eight to six. Figure 8.b. shows a cross-section diagram of one orbital plane. Notice that the 'LOS' lines cross through the earth's surface. Figure 8.b. is generated by clicking on the "Plot One Orbital Plane" button. Clicking on the "RF Crosslink Calculations" button will
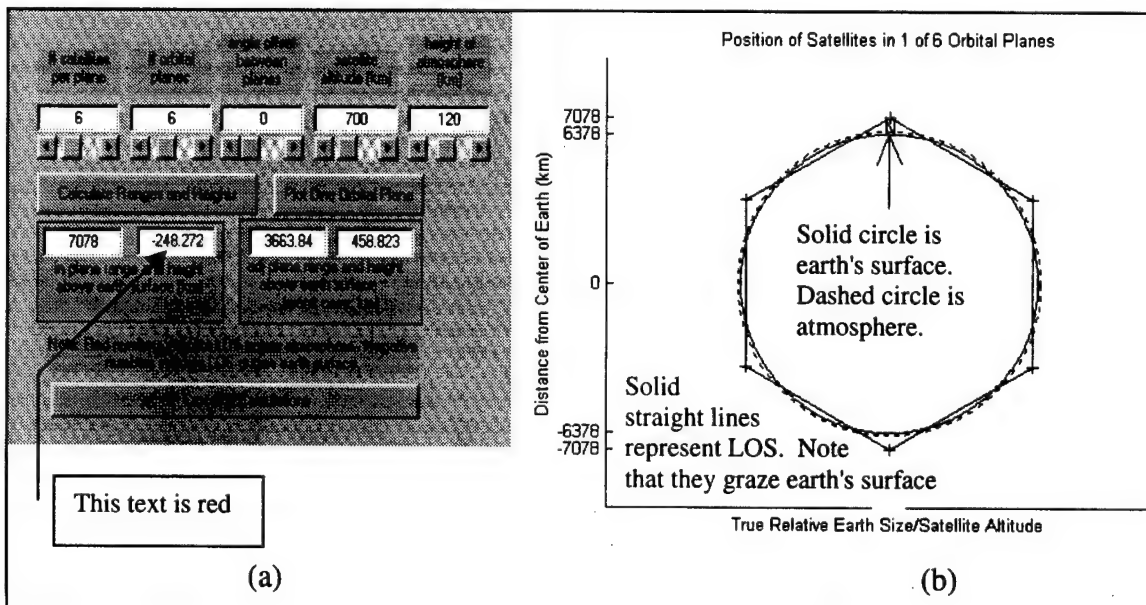
Figure 8. LOS Grazing Earth's Surface

open up the RF Tradeoffs GUI presented next. Appendix B lists the callback code for the Satellite Range Calculations GUI.

3.3 Radio Frequency Communications Tradeoffs GUI

The RF Tradeoffs GUI allows the user to select values for the communications system's parameters and solve for an unknown parameter. A solution for any unknown parameter can be made, given all other parameter values. The assumptions and equations used in this GUI's calculations were introduced in Section 2.3. This GUI is shown in Figure 9. This GUI can solve for any of the following main parameters: Range, E_b/N_o, R_D, M, lambda, P_t, L_l, eta, SNT, L_a, and Radius. The main parameters have radiobuttons (buttons with white circles to the left of the text label) labeled with their names on top and edit boxes below. The values in the main parameter edit boxes are used to compute

Figure 9. RF Tradeoffs GUI

the value of the unknown parameter. Values for the main parameters can be directly entered into the edit boxes. This is the only method to enter values for R_D, M, lambda, P_t, L_l, eta, and Radius. Lambda is a main parameter, but this GUI shows both wavelength and frequency for the user's convenience; entering a new value in either the lambda parameter edit box or the frequency parameter edit box automatically changes the value in the other edit box. Values for the Range parameter edit box can be imported from the Satellite Range Calculations GUI (see Section 3.3.1). Values for the E_b/N_o parameter edit box can be calculated based on other information entered in this GUI (see Section 3.3.2). Values for the SNT parameter edit box can be calculated based on other information entered in this GUI (see Section 3.3.3). Values for the L_a parameter edit box can be calculated based on other information entered in this GUI (see Section 3.3.4). The following sections describe the GUI's components and functionality in detail. Appendix C lists the callback code for the RF Tradeoffs GUI.

3.3.1 Imported and Default Values

When the RF Tradeoffs GUI is opened, the In-Plane-Range and Adjacent-Plane-Range values calculated by the Satellite Range Calculations GUI are automatically imported into their respective edit boxes at the top of the RF Tradeoffs GUI. Either of these ranges can be forwarded to the Range parameter edit box by clicking on (selecting) the appropriate radiobutton. The selected radiobutton shows a black dot in the circle; the radiobuttons are mutually exclusive. Clicking on the "RETURN TO RANGE CALCULATIONS" button closes the RF Tradeoffs GUI. Initially, all of the communication system's parameters edit boxes are blank. Values must be entered for all main parameters before solving for a main parameter (See Section 3.3.5). Clicking on

32

the "Reset All Default Values" button enters default values for all parameters (overwrites any existing values). Similarly, clicking on the "Reset default temps" button enters default values for the effective system noise temperature parameters. Table 2 lists the parameters, short descriptions, and the default values.

Table 2. Default Parameter Values

Parameter	Description	Default Value
Range	Range between satellites, km	7000 km
E_b/N_o	See Section 3.3.2	4.4 dB
Prob Bit Err	See Section 3.3.2	1e-5
Coding Gain	See Section 3.3.2	5.2 dB
Modulation	See Section 3.3.2	QPSK
R_D	Data rate, Mbps	512 Mbps
M	Link margin, dB	2 dB
lambda	Wavelength, m	0.005 m
freq	Frequency, GHz	60 GHz
P_t	Transmitter power, W	10 W
L_l	Antenna line and coupling losses, dB	1 dB
eta	Antenna efficiency (dimensionless)	0.55
SNT	See Section 3.3.3	34.87 dB-K
T_o	See Section 3.3.3	300 K
T_a	See Section 3.3.3	10 K
T_r	See Section 3.3.3	3000 K
L	See Section 3.3.3	1 dB
L_a	See Section 3.3.4	3 dB
TrackAcc	See Section 3.3.4	0.5
Radius	Antenna radius, m	0.5 m

3.3.2 Relationship Between E_b/N_o, P_b, Coding Gain, and Modulation

Equations 23, 24, 25, and 26 describe the relationships between E_b/N_o, P_b, coding gain, and the modulation scheme. Equations 23 and 25 show P_b as a function of E_b/N_o. However, most communications systems specify a value of P_b that determines (along with coding gain and modulation scheme) the value of E_b/N_o. Therefore, Equations 23 and 25 are inverted to show E_b/N_o as a function of P_b. However, Matlab® does not use the Q-function (Equation 24), it uses a closely related function called the complementary error function, erfc. The naming conventions for the Q-function and erfc are not consistent throughout literature. Sklar [Skl88] calls the Q-function (Equation 24) "the complementary error function or the co-error function" and calls Matlab's® erfc function "another form of the co-error function". The Q-function (Equation 24) is repeated here for reference. Matlab's® erfc function definition is given in Equation 29 and the relationship between the Q-function and Matlab's® erfc is given in Equation 30:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du \tag{24}$$

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp\left(-u^2\right) du \tag{29}$$

$$Q(x) = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right) \tag{30}$$

In addition, Matlab® uses functions called the error function, erf, and the inverse error function, erfinv:

$$erf(x) = 1 - erfc(x) \tag{31}$$

$$x = erfinv(y) \text{ satisfies } erf(x) = y \tag{32}$$

34

With this information, it is relatively straightforward to invert Equation 23, repeated here, which relates P_b and E_b/N_o for BPSK and QPSK:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \tag{23}$$

$$P_b = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_o}}\right) \tag{33}$$

$$2P_b = 1 - erf\left(\sqrt{\frac{E_b}{N_o}}\right) \tag{34}$$

$$\sqrt{\frac{E_b}{N_o}} = erfinv(1 - 2P_b) \tag{35}$$

$$\left.\frac{E_b}{N_o}\right|_{absolute} = [erfinv(1 - 2P_b)]^2 \tag{36}$$

$$\left.\frac{E_b}{N_o}\right|_{dB} = 10\log_{10}\left(\left.\frac{E_b}{N_o}\right|_{absolute}\right) \tag{37}$$

It is very important to note that Equation 36 finds the absolute ratio of E_b/N_o, not the more common decibel form of E_b/N_o. Inverting Equation 25, which relates P_b and E_b/N_o for M-ary PSK for $M > 2$ requires a bit more work. Assuming equality in Equation 25, as repeated here.

$$P_b = \frac{2Q\left(\sqrt{\frac{2kE_b}{N_o}}\sin\frac{\pi}{M}\right)}{k} \tag{25}$$

$$P_b = \left(\frac{1}{k}\right)erfc\left(\frac{\sqrt{\frac{2kE_b}{N_o}}\sin\frac{\pi}{M}}{\sqrt{2}}\right) \tag{38}$$

$$kP_b = 1 - erf\left(\frac{\sqrt{\frac{2kE_b}{N_o}}\sin\frac{\pi}{M}}{\sqrt{2}}\right) \tag{39}$$

$$\frac{\sqrt{\frac{2kE_b}{N_o}}\sin\frac{\pi}{M}}{\sqrt{2}} = erfinv(1 - kP_b) \tag{40}$$

$$\sqrt{\frac{2kE_b}{N_o}} = \left(\frac{\sqrt{2}}{\sin(\pi/M)}\right)erfinv(1 - kP_b) \tag{41}$$

$$\frac{2kE_b}{N_o} = \left[\left(\frac{\sqrt{2}}{\sin(\pi/M)}\right)erfinv(1 - kP_b)\right]^2 \tag{42}$$

$$\left.\frac{E_b}{N_o}\right|_{absolute} = \frac{1}{2k}\left[\left(\frac{\sqrt{2}}{\sin(\pi/M)}\right)erfinv(1 - kP_b)\right]^2 \tag{43}$$

$$\left.\frac{E_b}{N_o}\right|_{dB} = 10\log_{10}\left(\left.\frac{E_b}{N_o}\right|_{absolute}\right) \tag{44}$$

Equations 37 and 44 find a value for E_b/N_o that does not include the effects of coding gain. To find the coded E_b/N_o value required by the communications system, subtract the coding gain from the uncoded E_b/N_o value found in Equation 37 or 44.

In the RF Tradeoffs GUI, the coded E_b/N_o value required by the communications system is the value that is shown in the E_b/N_o parameter edit box. This value can be recalculated by directly entering a new value in the Prob Bit Err (P_b) or coding gain edit boxes and clicking on a modulation scheme radiobutton. Clicking on a modulation scheme radiobutton is required to initiate the recalculation of the coded E_b/N_o value. The radiobuttons are mutually exclusive. A value may be directly entered into the E_b/N_o parameter edit box, but this does not affect the values for P_b, coding gain, and the modulation scheme. The coded E_b/N_o is essentially a function of three variables: Pb, coding gain, and the modulation scheme. If only the coded E_b/N_o value is given, it is

impossible to conclusively solve for the three variables, because the given information is one equation with three unknowns. For a given coded E_b/N_o value, there are multiple combinations of P_b, coding gain, and the modulation scheme that can yield the coded E_b/N_o value.

3.3.3 Relationship Between SNT, T_o, T_a, T_r, and L

Equation 16, repeated here, shows effective system noise temperature as a function of ambient temperature (T_o), antenna noise temperature (T_a), the noise temperature of the front-end receiver (T_r), and receive line and coupling losses (L):

$$T = T_a/L + (L-1)T_o/L + T_r \tag{16}$$

Equation 16 has units of absolute Kelvin temperature. Equation 45 shows the decibel form of effective system noise temperature (SNT) used by the RF Tradeoffs GUI:

$$SNT = 10\log_{10}(T_a/L + (L-1)T_o/L + T_r) \tag{45}$$

Directly entering a new value into the T_o, T_a, T_r, or L parameter edit boxes automatically recalculates a new value of SNT. A value may be directly entered into the SNT parameter edit box, but this will not affect values for T_o, T_a, T_r, or L. SNT is a function of four variables; if only SNT is given, it is impossible to conclusively solve one equation with four unknowns. This GUI cannot solve for the separate component parameters of SNT (T_o, T_a, T_r, or L) when given SNT and three of the four parameters. Figure 10 contains four plots showing the behavior of Equation 45 as a function of each of the component parameters. On each plot, the parameter of interest is varied and the other three component parameters are set to their default values. The asterisks on each plot indicate the default value of the parameter of interest. With all component parameters set to their default values, SNT equals 34.87 dB-K. Figure 10 shows that if SNT and three of

37

the four component parameters are given, the value of the fourth parameter may be impractical or impossible. From Figure 10.d., if SNT is given as 35.2 dB-K and T_o, T_a, and T_r are set to their default values, the expected value of L would be impracticably large because of the asymptotic behavior of the curve. From Figure 10.a., if SNT is given as 34.6 dB-K and T_a, T_r, and L are set to their default values, the expected value of T_o
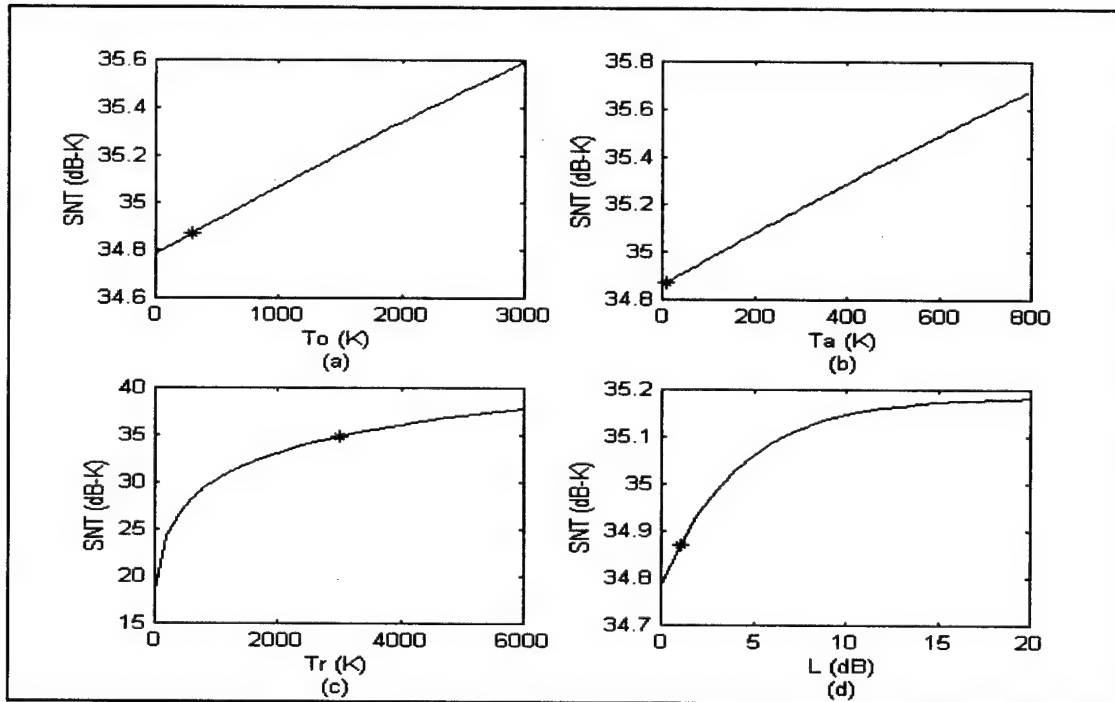
Figure 10. SNT as a Function of Component Parameters

would appear to be negative, which is impossible for Kelvin temperatures. To prevent problems and errors with these calculations, this GUI cannot solve for the individual component parameters of SNT.

3.3.4   Relationship Between L_a and Tracking Accuracy

Equation 13 shows pointing loss (L_a) as a function of tracking accuracy and the reciprocal of the half-power beamwidth of the antenna.   If the tracking accuracy

(TrackAcc) is expressed as a fractional amount of the half-power beamwidth, the half-power beamwidth components of Equation 13 cancel and it simplifies to:

$$L_a = 12(TrackAcc)^2 \qquad (46)$$

Directly entering a new value into either of the L_a or TrackAcc parameter edit boxes automatically recalculates the other value. The GUI cannot solve directly for TrackAcc given the other main parameters.

3.3.5 Solving for an Unknown Parameter Given All Other Parameters

The RF Tradeoffs GUI can solve for any of the main parameters when given values for all of the other main parameters. To select which 'main' parameter the GUI will solve for, click on the appropriate parameter's radiobutton. As mentioned earlier, the radiobuttons are mutually exclusive. Clicking on a parameter's radiobutton causes the parameter name to appear below the "ENTER ALL OTHER PARAMETERS AND COMPUTE" words and button. In addition, the text color of the selected parameter's edit box changes to white. This causes the value in the selected parameter's edit box to visually disappear, but the GUI can still process the value. A value must have been in the selected parameter's edit box prior to clicking on the parameter's radiobutton to prevent the GUI from returning an error. As described in Section 3.1, the GUI will return an error if an empty string is entered. When the "Reset All Default Values" button is clicked, it enters 512 (Mbps) into the R_D parameter edit box, and then selects to solve for R_D, thus turning the R_D parameter edit box text white. When a different parameter's radiobutton is selected, the text color of the just de-selected parameter's edit box returns to black and the previous value (the value in the edit box prior to selecting that parameter) reappears.

39

Clicking on the "COMPUTE" button performs the computation to solve for the unknown parameter. The result of the computation is displayed in the edit box below the "ENTER ALL OTHER PARAMETERS AND COMPUTE" words and button next to the name of the selected parameter. This result is not automatically forwarded into its parameter edit box when a different parameter radiobutton is selected. For example, assume the GUI is solving for Range and the result is 4239 km. Suppose the user wants to use this range value in the GUI to solve for a different parameter such as eta. When the user clicks on eta's radiobutton, the Range parameter edit box restores its previous value (as described in the last paragraph), not 4239. To use 4239, the user must directly enter 4239 into the Range parameter edit box. The actual calculation is based on the link equation. Using a modified form of Equation 19 and Equation 22:

$$0 = P_t - E_b/N_o - R_D - M - L_l + 20\log(4\pi^2R^2\eta/\lambda^2)$$

$$- 20\log(4\pi r/\lambda) - L_a + 228.6 - SNT \qquad (47)$$

The formula for effective system noise temperature (SNT) is given in Equation 45. To solve for P_t, E_b/N_o, R_D, M, L_l, L_a, or SNT given the other parameters, the desired parameter is brought over to the left side of Equation 47 and the given values are substituted in. To solve for Range (r), eta (η), lambda (λ), or Radius (R) given the other parameters, the process is the basically the same, but more computations are required to eliminate the logarithm term.

Clicking on the "GO TO GRAPHS" button performs the same actions as clicking on the "COMPUTE" button and also opens the RF Graphs GUI. The RF Graphs GUI (Section 3.4) visually shows the trade-off between any two parameters.

3.3.6 Calculate System Parameters (FSL, Gain, 3dB Beamwidth, G/T)

Clicking on the "COMPUTE" button solves for the desired parameter and also calculates the following system parameters: Free Space Loss (FSL), Gain, 3dB Beamwidth, and the G/T figure of merit. The FSL calculation is based on Equation 11. Note that FSL is shown as a negative number. The system's Gain calculation is based on Equation 9. The 3dB beamwidth of the antenna calculation is based on Equation 12. Finally, the G/T figure of merit calculation is based on Equation 18. These system parameters are mathematical functions of different main parameters and provide useful comparisons between different systems.

$$L_{FSL} = -20\log(4\pi r/\lambda) \tag{48}$$

$$Gain = 10\log(4\pi^2 R^2 \eta/\lambda^2) \tag{49}$$

$$BW_{3dB} = 35\lambda/R \tag{50}$$

$$G/T = 10\log(4\pi^2 R^2 \eta/\lambda^2) - SNT \tag{51}$$

3.3.7 Invalid Computation Results

The RF Tradeoffs GUI solves for a desired parameter when values for all other main parameters are given. However, sometimes the result of the computation yields a physically impossible value for the desired parameter. If this occurs, the invalid parameter value is displayed in red. This error checking is performed only on the parameter of interest. It is not performed on the given parameters being used to solve for the desired parameter. The parameters that are checked and their associated invalid values are listed in Table 3.

41

Table 3. Invalid Parameter Values

Parameter	Min Value	Max Value	Description
Eb/No	-1.59		Shannon Limit
Ll	0		Losses are defined as positive quantities that are subtracted in equations. A negative loss would be a gain.
La	0		
eta	0	1	Efficiency defined 0 to 1
M	0		Margin < 0 indicates no communication possible
Radius	0		Antenna physical radius must be > 0

3.4 Radio Frequency Communications Graphs GUI

The RF Graphs GUI allows the user to graph any one parameter against any other parameter. This allows the user to visualize the trade-off between the two parameters. The assumptions and equations used in this GUI's calculations are an extension of those developed by the RF Tradeoffs GUI. The RF Graphs GUI is shown in Figure 11. The following sections describe the GUI's components and functionality in detail. Appendix D lists the callback code for the RF Graphs GUI.

3.4.1 Imported Values

The RF Graphs GUI automatically imports the values of the main parameters (Range, E_b/N_o, R_D, M, lambda/frequency, P_t, L_l, eta, SNT, L_a, and Radius) and system parameters (FSL, Gain, 3dB Beamwidth, and G/T) from RF Tradeoffs GUI. These values are listed in edit boxes on the left side of this GUI. By default, this GUI displays frequency. The display can alternate between frequency and wavelength by using the drop-down box selection. The imported values are used as reference values for the graphing. New values for any of the parameters are directly entered into the edit boxes. Entering a new value for Range, lambda/frequency, eta, SNT, or Radius recalculates the
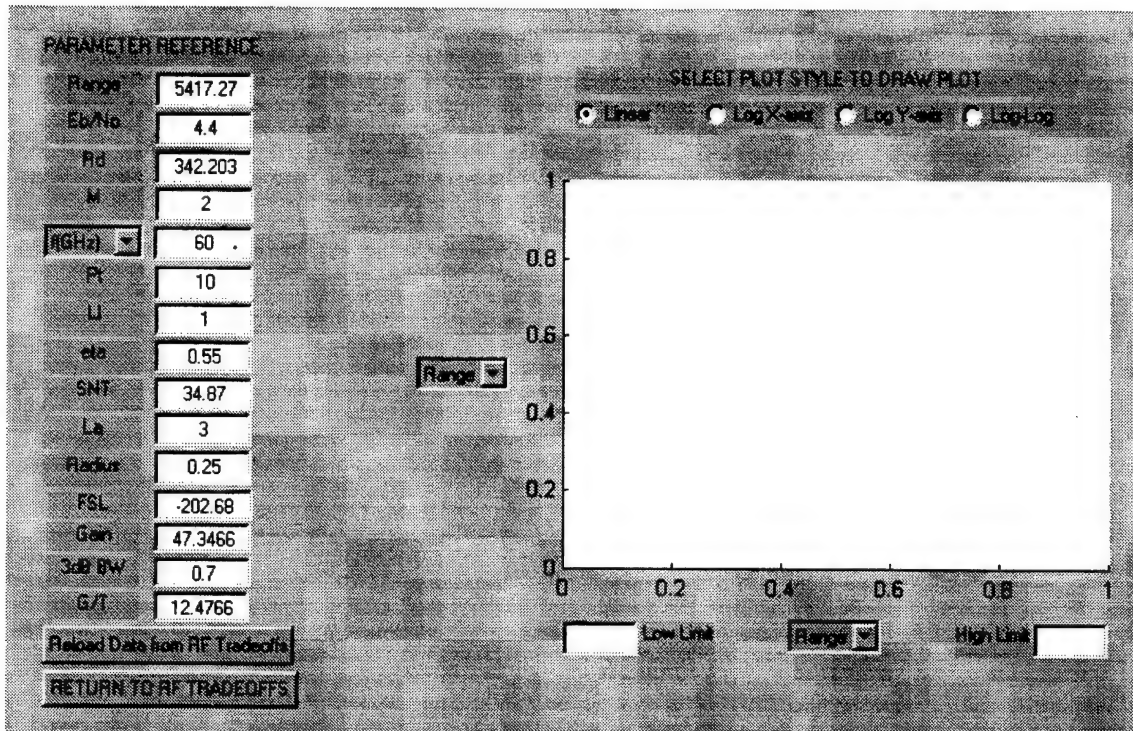
Figure 11. RF Graphs GUI

system parameters. Clicking on the "Reload Data from RF Tradeoffs" button resets all parameters to the initially imported values. Clicking on the "RETURN TO RF TRADEOFFS" button closes the RF Graphs GUI.

3.4.2 Selecting Parameters to Plot

The plotted parameters on each axis are selected from the drop-down boxes located along the axes of the plot. The x-axis displays any of the main parameters. The y-axis displays any of the main parameters or system parameters. Each axis can display either wavelength or frequency. Choosing parameters to be plotted allows the trade-off between the two parameters to be visualized. The same main parameter can be selected for plotting on both axes, but the resulting y = x plot is trivial. When the y-axis is selected to display one of the main parameters, the trade-off plot should not be interpreted as a function in the classical mathematical sense. Eta, antenna efficiency,

43

does not depend upon range. However, with eta selected on the y-axis and Range selected on the x-axis, the trade-off plot (Figure 12.a.) shows that the communication system's range increases as the antenna efficiency increases. This relationship is based on the link equation (Equation 47), with all other main parameters held constant. When the y-axis is selected to display one of the system parameters (FSL, Gain, 3dB Beamwidth, and G/T) the trade-off plot shows a functional relationships in the classic mathematical sense. FSL is a function of Range and lambda. When FSL is selected on the y-axis and any main parameter except Range, lambda, or frequency is selected on the x-axis, the trade-off plot is a constant (horizontal line) at the value shown in the FSL parameter edit box. When FSL is selected on the y-axis and Range is selected on the x-axis, the trade-off plot (Figure 12.b.) shows that FSL increases in magnitude as Range increases. This functional relationship is based on Equation 48.
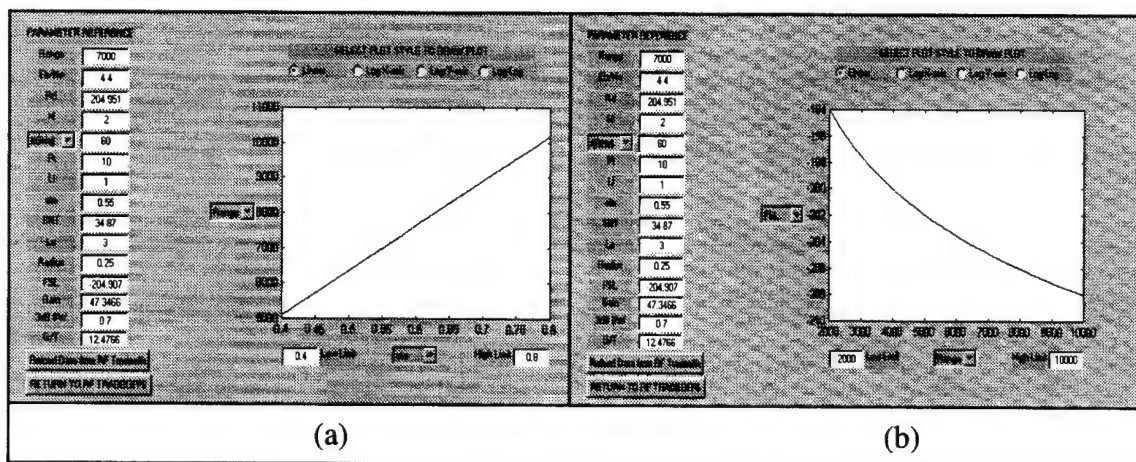
Figure 12. Range/Eta and FSL/Range Plots

3.4.3 Setting X-Axis Range

When a main parameter is selected for the x-axis, the RF Graphs GUI automatically pre-sets the lower and upper limits of the x-axis. Table 4 lists the parameters and the limits. For the parameters listed in Table 3 (Invalid Parameter Values), the pre-set values

44

of the parameters are within the range of valid values. The limits are changed by directly entering a new value in either the Low Limit or High Limit edit boxes. If the value entered in the Low Limit edit box is greater than the value entered in the High Limit edit box, this GUI will automatically switch the values when plotting. The plotting routine always displays the x-axis values in ascending order from left to right. An invalid parameter value (according to Table 3) can be directly entered into either the Low Limit or High Limit edit boxes.

Table 4. Preset X-Axis Limits

Parameter	Lower Limit	Upper Limit
Range, km	2000	10000
Eb/No, dB	2	12
RD, Mbps	10	1000
M, dB	0.5	4.5
lambda, m	0.001	0.1
frequency, GHz	3	300
Pt, W	1	50
Ll, dB	0.5	3
eta, dimensionless	0.4	0.8
SNT, dB-K	33	37
La, dB	0.5	3
Radius, m	0.1	1

3.4.4 Displaying the Plots

Plots are computed and displayed by clicking on any one of the plot style radiobuttons. The choices for plot style are Linear, Log X-axis, Log Y-axis, and Log-Log. The Linear style plots both the x-axis and y-axis with a linear scale. The Log X-axis style plots the x-axis with a logarithmic scale and the y-axis with a linear scale. The

Log Y-axis style plots the x-axis with a linear scale and the y-axis with a logarithmic scale. The Log-Log style plots both the x-axis and y-axis with a logarithmic scale. Prior to selecting a plot style radiobutton, the parameters for the x-axis and y-axis must have been selected using the drop-down boxes and the x-axis Low Limit and High Limit edit boxes must contain values (pre-set or directly entered). With a plot displayed, clicking on a different plot style radiobutton causes a recalculation of the plot with the new style, using the same parameters for the x-axis and y-axis and the same Low Limit and High Limit for the x-axis. The plotting routine uses the link equation (Equation 47) to calculate the y-axis parameter values given the x-axis parameter values at the Low Limit, the High Limit, and nineteen points evenly linearly spaced between the Low Limit and High Limit. The low and high limits for the y-axis are automatically set based on the results of the calculations. The y-axis limits cannot be manually changed. The plotting routine always displays the y-axis values in ascending order from bottom to top. Since FSL is defined as a negative number, this causes an FSL versus Range plot to slope downwards because FSL increases in magnitude as Range increases (see Figure 12.b.). The Log Y-axis and Log-Log plot styles cannot display negative y-axis values because the logarithm of a negative real number is complex. For plots where the y-axis parameter has both positive and negative values over the x-axis parameter range, the Log Y-axis and Log-Log plot styles will not display the plot where the y-axis parameter values are negative. See Figure 13. Figure 13.a. is a linear plot of L_l versus Range. The calculated value of L_l changes from positive to negative at approximately 6000 km. Figure 13.b. is a Log Y-axis style plot. The plot is not displayed above approximately 6000 km. (The triangles on Figure 13.a. are discussed in Section 3.4.5.) When FSL is selected as the

46
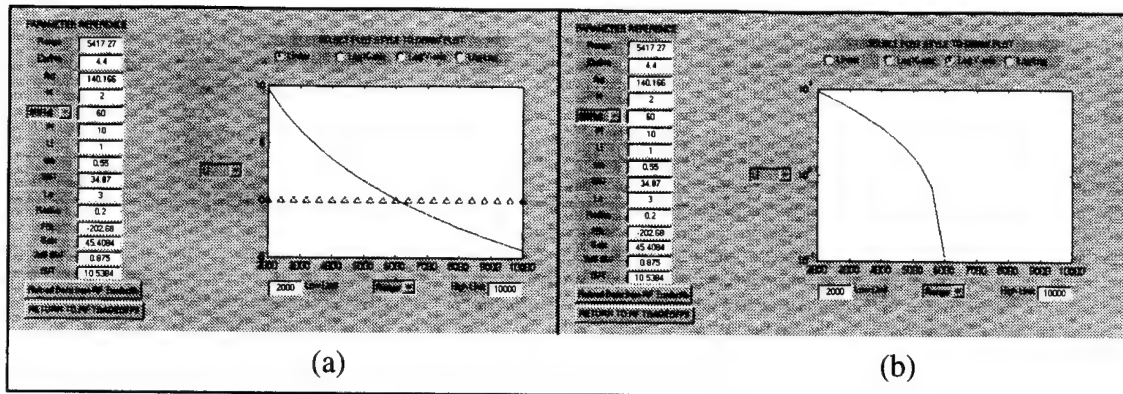
Figure 13. Logarithmic Y-Axis with Negative Y-Axis Values

y-axis parameter, selecting the Log Y-axis and Log-Log plot style radiobuttons has no effect (i.e., the radiobuttons can be selected, but the previous plot remains).

3.4.5 Invalid Parameter Values

If the y-axis parameter selection is E_b/N_o, L_l, L_a, eta, M, or radius, certain plot styles show the valid limits of the parameter, according to Table 3. The limit is denoted by a horizontal line of triangles at the y-axis value corresponding to the limit. For minimum valid limits, the triangles point upwards, pointing towards valid values for the y-axis parameter. For maximum valid limits, the triangles point downwards, pointing towards valid values for the y-axis parameter. Eta is the only parameter with a maximum valid limit. See Figure13.a. and Figure 14 for plots with the valid limit indicators. The minimum valid limit indicators will be displayed when the Linear or Log X-axis plot styles are selected. The minimum valid limit indicators can not be displayed when the Log Y-axis or Log-Log plot styles are selected. The minimum valid limits are either zero or negative; the logarithm of zero is negative infinity and the logarithm of a negative real number is complex. The maximum valid limit is displayed with all plot styles.

47
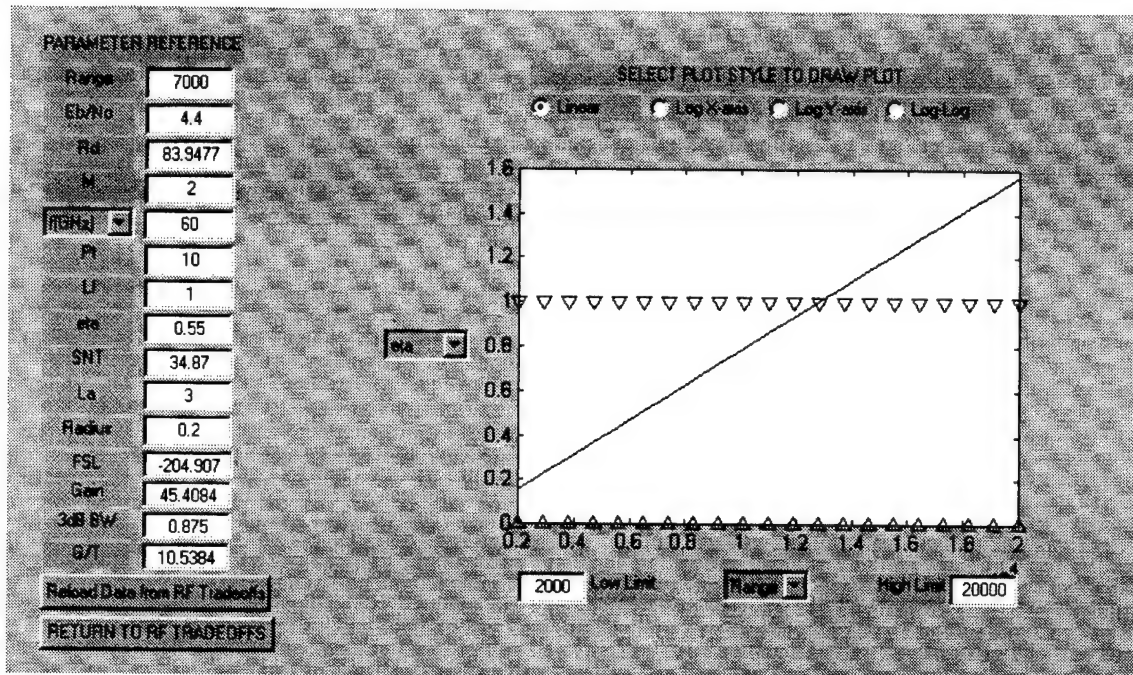
Figure 14. Minimum and Maximum Valid Limit Indicators

3.5 Summary

This chapter introduced the three GUIs used to analyze the ISLs. The Satellite Range Calculations GUI calculated the ranges between satellites in a constellation. It also calculated the minimum LOS height and visually indicated when the LOS passed through the atmosphere or grazed the earth. The RF Tradeoffs GUI modified the link equation introduced in Chapter 2 to solve for any unknown parameter of the link equation. It also computes system parameters (FSL, Gain, 3dB beamwidth, and G/T) to allow comparison of different RF systems. The mathematics underlying the relationship between E_b/N_o, P_b, coding gain, and modulation was developed in detail for use in the RF Tradeoffs GUI. The RF Graphs GUI extended the link equation used in the RF Tradeoffs GUI to allow visualization of the trade-offs between two parameters in the link equation.

48

4 Results and Analysis

4.1 Introduction

This chapter examines "interesting results" from each of the GUIs. "Interesting results" is an arbitrary term, but the goal is to analyze sets of parameters which are limiting cases or identify significant factors in calculations. The GUIs operate in real time, so the parameters that yield the "interesting results" can be easily modified and analyzed further.

4.2 Results/Analysis of Satellite Range Calculations GUI

The Satellite Range Calculations GUI presents limiting cases when the LOS travels through the atmosphere or grazes the earth's surface. Communications that travel through the atmosphere experience fading and diffraction effects. A LOS that grazes the earth's surface is obviously blocked. The GUI indicates these limiting cases by displaying the minimum LOS height with red text. Table 5 lists representative cases for the Satellite Range Calculations GUI. In all cases, the height of the atmosphere parameter is set to its default value of 120 km. As a first step, a satellite altitude is arbitrarily selected. The

Table 5. Satellite Range Calculations GUI Limiting Cases

Altitude	Min # sats per plane	In-plane range	Min in-plane height	Min # orbital planes	Offset angle	Adj-plane range	Min adj-plane height
700	8	5417.3	161.2	5	0	4374.4	353.6
					22.5	4734.9	292.3
770	8	5470.9	225.9	4	0	5470.9	225.9
					22.5	5921.6	128.0
850	7	6272.2	134.2	4	0	5532.1	300.0
					25.714	6140.1	165.5
1000	7	6402.4	269.3	4	0	5646.9	438.4
					25.714	6267.6	301.4

altitudes listed in Table 5 are low-earth orbits. Then, the minimum number of satellites per plane and the minimum number of orbital planes are found by empirical analysis. These are the minimum values resulting in a LOS that does not enter the earth's atmosphere. The range and minimum LOS height for satellites in the same orbital plane are shown in Table 5. The range and minimum LOS height for satellites in adjacent orbital planes are shown for two offset angles. The first offset angle is zero and the second offset angle is the optimum phasing angle, which is a function of the number of satellites in each orbital plane. For example, if there are eight satellites in an orbital plane, they are spaced every 45 degrees. The optimum phasing angle is half the spacing, or 22.5 degrees. The optimum phasing angle provides the best coverage of the earth's surface for a given satellite constellation geometry [AdR87]. This GUI only calculates ranges and heights; it does not provide information about coverage or potential gaps in coverage.

Table 6 lists information for a satellite constellation of six orbital planes with eight satellites in each plane. A forty-eight satellite constellation is representative of the number of satellites in the proposed Discoverer II constellation. As with Table 5, the height of the atmosphere is 120 km and the same satellite altitudes are used. The 48-

Table 6. Ranges/Heights for 8 Satellites per Plane, 6 Planes

Altitude	In-plane range	In-plane height	Offset angle	Adj-plane range	Adj-plane height
700	5417.3	161.2	0	3663.8	458.8
			22.5	3965.7	416.6
770	5470.9	225.9	0	3700.1	526.4
			22.5	4005.0	483.8
850	5532.1	300.0	0	3741.5	603.7
			22.5	4049.8	560.6
1000	5646.9	438.4	0	3819.1	748.6
			22.5	4133.8	704.6

satellite geometry is larger than the limiting cases at all altitudes. As the altitude increases, the in-plane ranges and heights increase. For the adjacent-plane information, the ranges and heights increase as altitude increases, given the same offset angle. In all cases, the in-plane range is greater than the adjacent-plane range (for either offset angle).

4.3   Results/Analysis of the Radio Frequency Communications Tradeoffs GUI

The RF Tradeoffs GUI shows limiting cases for the unknown parameter (refer to Section 3.3.7). The GUI indicates when an invalid value is calculated for the unknown parameter by displaying it with red text. The following two sections analyze the E_b/N_o main parameter and evaluate significant factors in the RF Tradeoffs GUI.

4.3.1   Analysis of E_b/N_o Parameter

The E_b/N_o main parameter shows the amount of coded E_b/N_o required by the system. Equation 21 calculates the ratio of carrier energy-to-noise energy density (C/N_o) available at the receiver. This available C/N_o is "supplied" to the demodulation and decoding processes (see Figure 4). The demodulation and decoding processes require a certain amount of C/N_o. Equation 22 shows this requirement:

$$C/N_o = E_b/N_o + R_D + M \tag{22}$$

If the required C/N_o (Equation 22) exceeds the available C/N_o (Equation 21), the communications link cannot be established. The E_b/N_o term in Equation 22 represents the uncoded E_b/N_o required by the system. Assuming C/N_o and data rate (R_D) are held constant, Equation 22 shows that increasing the required E_b/N_o lowers the link margin.

In the RF Tradeoffs GUI, the E_b/N_o main parameter is directly entered or calculated by entering values for probability of bit error, coding gain, and selecting a modulation scheme. BPSK and QPSK modulation have the same probability of bit error and

51

therefore produce identical calculations. This is based on Equations 23 and 37 and is shown graphically in Figure 5. The BPSK and QPSK modulation schemes have separate radiobuttons, but the GUI uses the same code callback for both radiobuttons.

4.3.2 Significant Factors

The significance of any main parameter can be evaluated using the RF Tradeoffs GUI. Set up the GUI to solve for any other main parameter, and evaluate how that parameter responds to changes in the "significant parameter." In other words, evaluate the change in output with respect to a change in input. Table 7 lists the results of a significance test. The GUI solved for Margin; with the other main parameters set to the values shown in the Original Parameter column, the original margin calculation was 12.3191 dB. (Four decimal places is Matlab's® default numeric format.) Each main parameter was individually changed to the value shown in the New Parameter column,

Table 7. Significant Factors in RF Tradeoffs GUI

Parameter	Original Parameter	Original Margin	New Parameter	New Margin	% Change
Range	5400 km		6400	10.8434	-11.98
E_b/N_o	4.4 dB		5.4	11.3191	-8.12
R_D	512 Mbps		1024	9.3088	-24.44
lambda	60 GHz		23.5	4.1774	-66.09
P_t	10 W	12.3191 dB	20	15.3294	24.44
L_l	1 dB		3	10.3191	-16.24
eta	0.55		0.7	14.4138	17.00
SNT	34.87 dB-K		36.87	10.3191	-16.24
L_a	3 dB		0.75	14.5691	18.26
Radius	0.5 m		0.25	0.2779	-97.74

and the margin was recalculated. The New Margin value and percent change from the Original Margin value are shown. Analysis of Table 7 shows interesting results. E_b/N_o, L_l, SNT, and L_a are traded-off with Margin on a one-for-one basis. For example, increasing the required E_b/N_o from 4.4 dB to 5.4 dB decreases the margin from 12.3191 dB to 11.3191 dB. This result follows from analysis of the link equation (Equation 47). The changes in R_D and P_t have the same magnitude effect on margin. The parameter R_D has units of megabits per second and P_t has units of watts. Both units are converted to decibel form for use in the link equation (Equation 47). The New Parameter values for both R_D and P_t are twice their original values. A factor of two change in absolute numbers corresponds to a 3 dB change (the GUI calculates a 3.0103 dB change). Numerically, the most significant change in margin occurs when the antenna radius changes from 0.5 to 0.25 m. Because the main parameters have different units, Table 7 does not always represent an "apples to apples" comparison. However, antenna radius appears to be a significant factor in determining the capability of the communications system and physical antenna size is usually a very important spacecraft design consideration.

4.4   Results/Analysis of Radio Frequency Communications Graphs GUI

The RF Graphs GUI provides the same basic analysis capability as the RF Tradeoffs GUI. However, the RF Graphs GUI can plot any one of fifteen y-axis parameters and any one of eleven x-axis parameters, for a total of 165 separate plots. The data visualization of the graphs is more powerful than analysis of a single number because the graphs show trends over a range of data.

4.4.1 Parametric Trade-Off Analysis Capability

The RF Graphs GUI displays the reference values of the parameters on the left side of the GUI. When two parameters are selected for plotting, the other parameter reference values are used in the link equation (Equation 47) to compute the plot. Changing the value of one of the reference parameters shows the trade-off plot with respect to the value of that reference parameter. Figure 15 shows plots of Range versus Radius for Power values of 10, 20, and 50 W. Because of space limitations, the entire RF Graphs GUI is

(a) 10 W   (b) 20 W

(c) 50 W
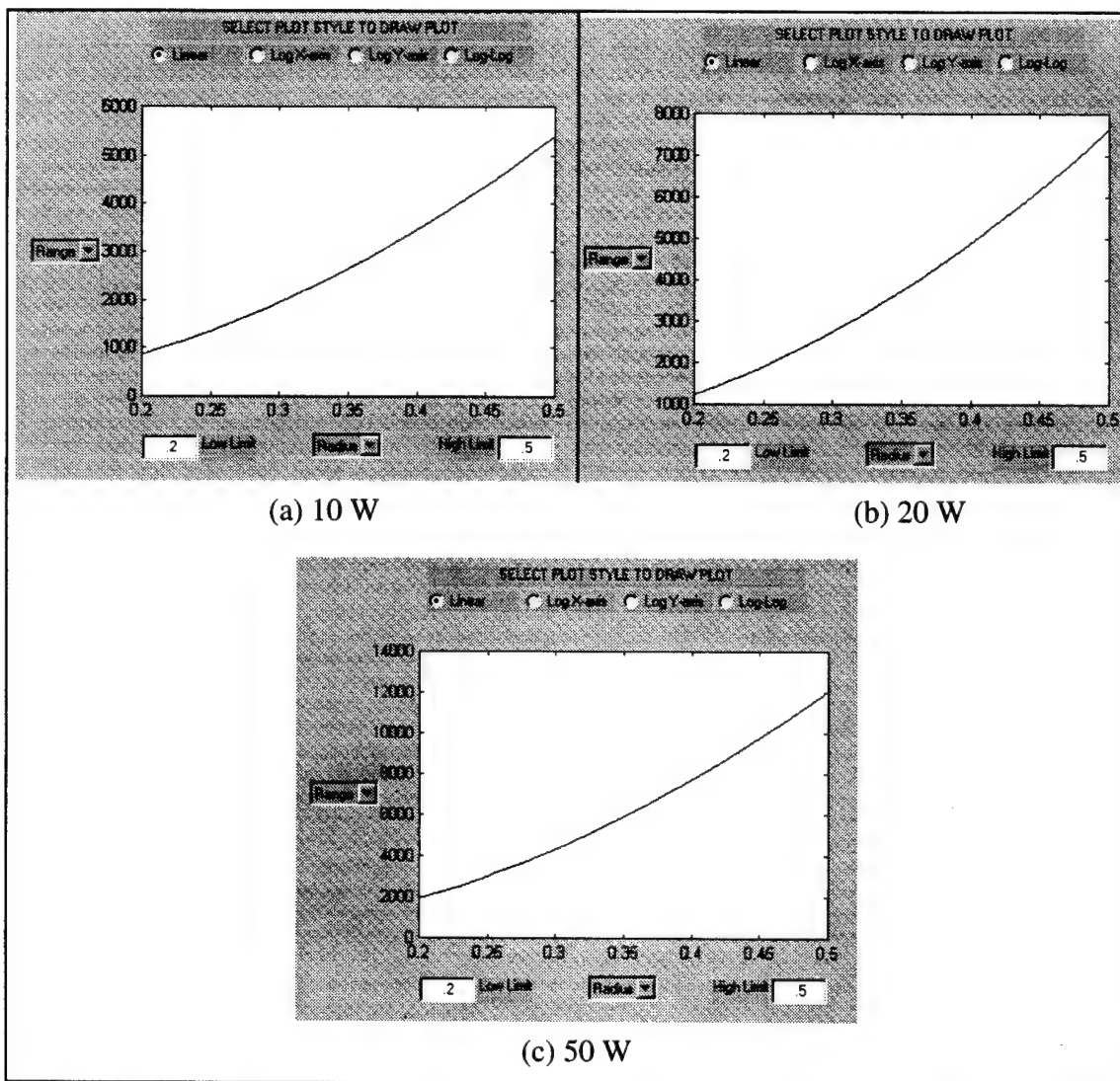
Figure 15. Range vs. Radius as Power Varies

not shown in Figure 15. The RF Tradeoffs GUI is used to set the parameter values according to the Original Parameter column of Table 7. The RF Tradeoffs GUI can be used to double-check the plot values in Figure 15. To begin, set the RF Tradeoffs GUI to solve for Range and directly enter 12.3191 into the Margin parameter edit box. Compute the Range when Power is set to 10, 20, and 50 W and the Radius is set to 0.2 and 0.5 m. These results are shown in Table 8. These values correspond to the endpoints of the plots in Figure 15.

Table 8. Range Values Given Radius and Power

Power	Radius = 0.2	Radius = 0.5
10	864	5400
20	1222	7636
50	1932	12074
Note: Range values truncated to integer values		

Figure 15 is composed of a separate plot for each Power value. The RF Graphs GUI clears the screen before each plot, so multiple plots cannot be displayed on the screen simultaneously. A 'multiple plot' feature is a possible improvement to this GUI, as discussed in Chapter 5.

4.4.2 Significant Trade-Off Plots

Some parameters in an ISL are more significant than other parameters. It follows that the trade-offs between significant parameters are more important than the trade-offs between other parameters. Range is a significant parameter because it drives free space loss, which is the largest single loss factor in the link equation. The satellite constellation design directly affects range. Radius is a significant parameter because the physical size

55

and weight of the antenna is a major factor in satellite design. Transmitter power is a significant parameter because the transmitter's power consumption, weight, and size are major factors in satellite design. Data rate (R_D) is a significant parameter because it is commonly specified as a requirement. A communications system may be required to maintain a specific data rate to communicate in real time or because of data formatting requirements. Antenna efficiency, eta, is a significant factor in that the satellite design can impose constraints on the antenna type, size, or construction, all of which can affect the antenna efficiency. Figures 16 through 25 plot significant trade-offs. This thesis uses two conventions when discussing the plots. First, a plot of "Parameter 1" vs. "Parameter 2" displays Parameter 1 on the y-axis and Parameter 2 on the x-axis. Second, Parameter 2 on the x-axis is considered the independent variable and Parameter 1 on the y-axis is considered the dependent variable. The reference parameters used in Figures 16 through 25 are listed in Table 9; these values satisfy Equation 47. Figures 16 through 20 plot the

Table 9. Reference Parameters for Figures 16-25

Parameter	Reference Value
Range	5400 km
E_b/N_o	4.4 dB
R_D	512 Mbps
Margin	12.3191 dB
lambda	60 GHz
P_t	10 W
L_l	1 dB
eta	.55
SNT	34.87 dB-K
L_a	3 dB
Radius	.5 m
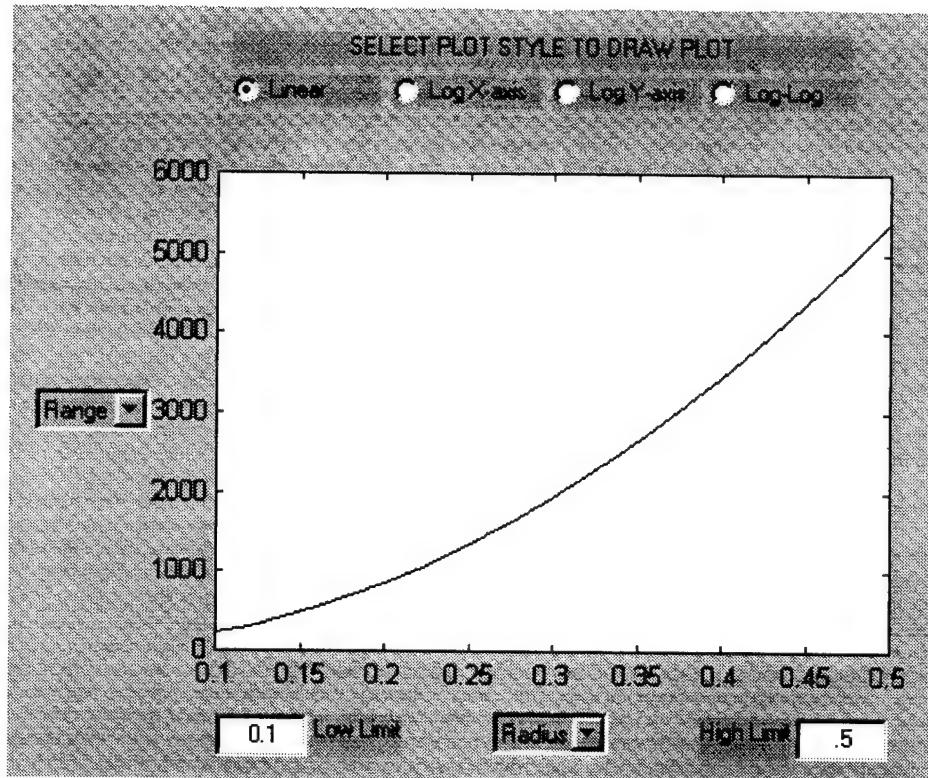
Figure 16. Range vs. Radius (Linear)
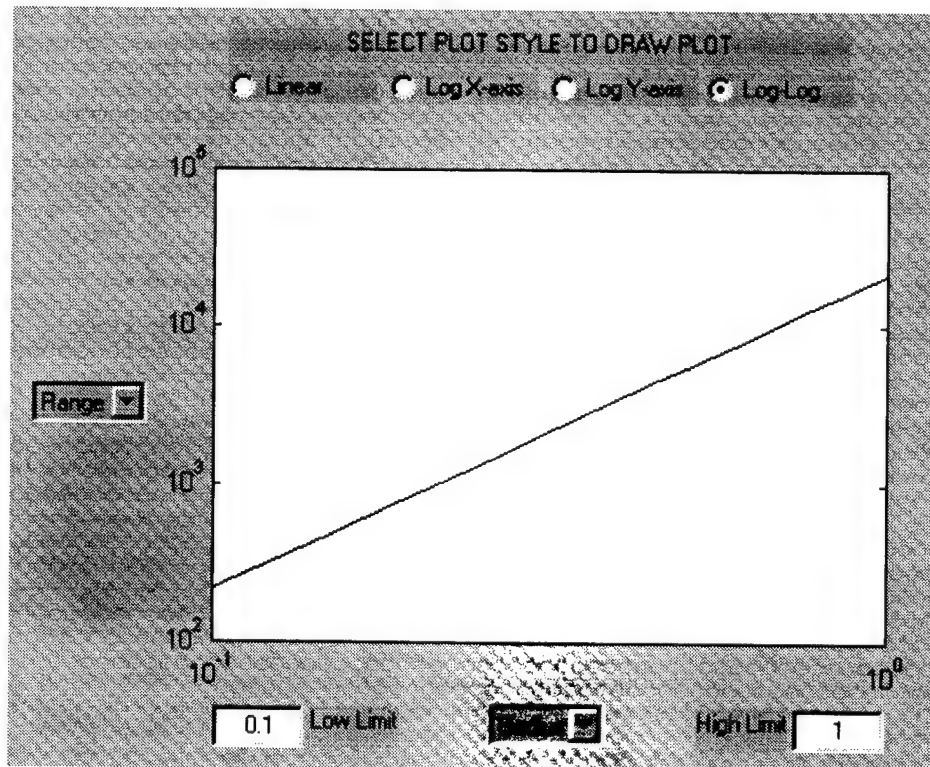
Figure 17. Range vs. Radius (Log-Log)
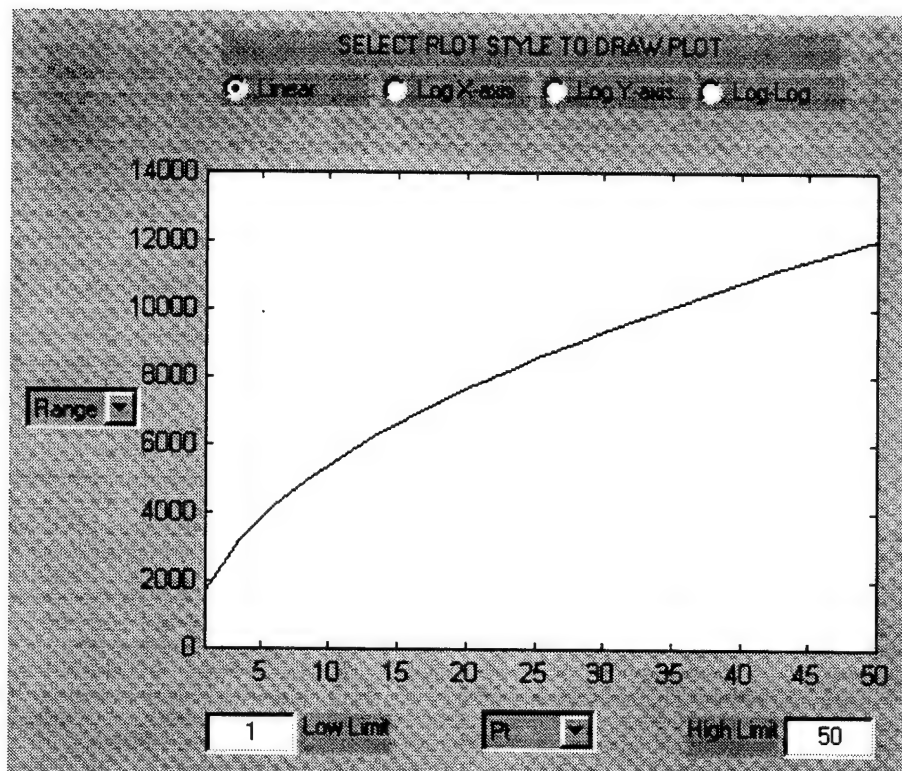
Figure 18. Range vs. Power (Linear)
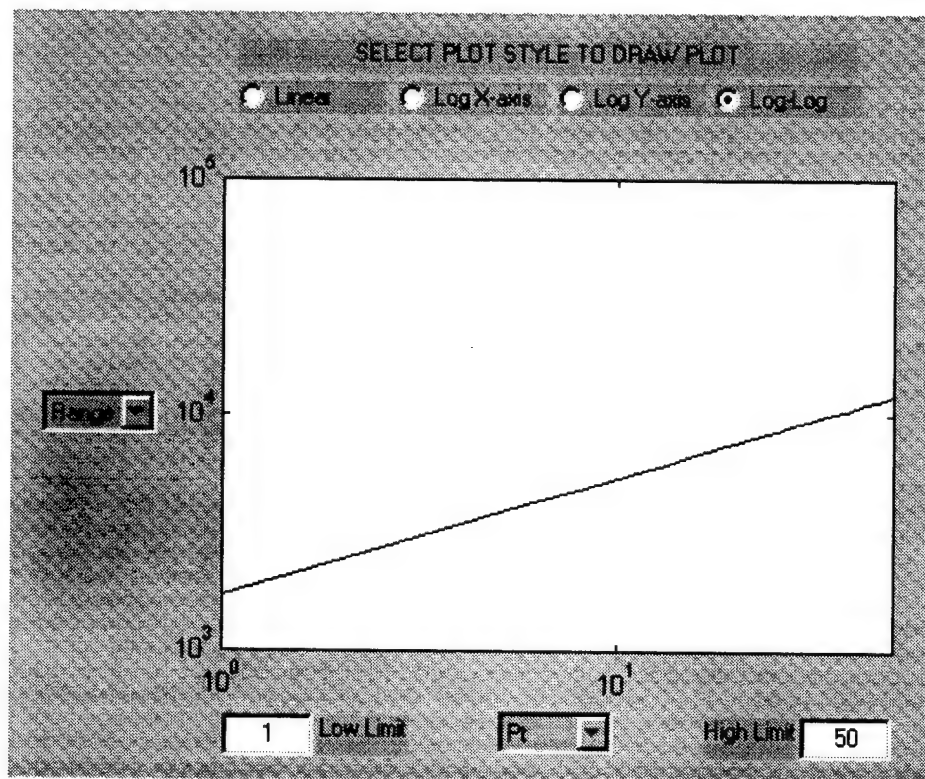
Figure 19. Range vs. Power (Log-Log)
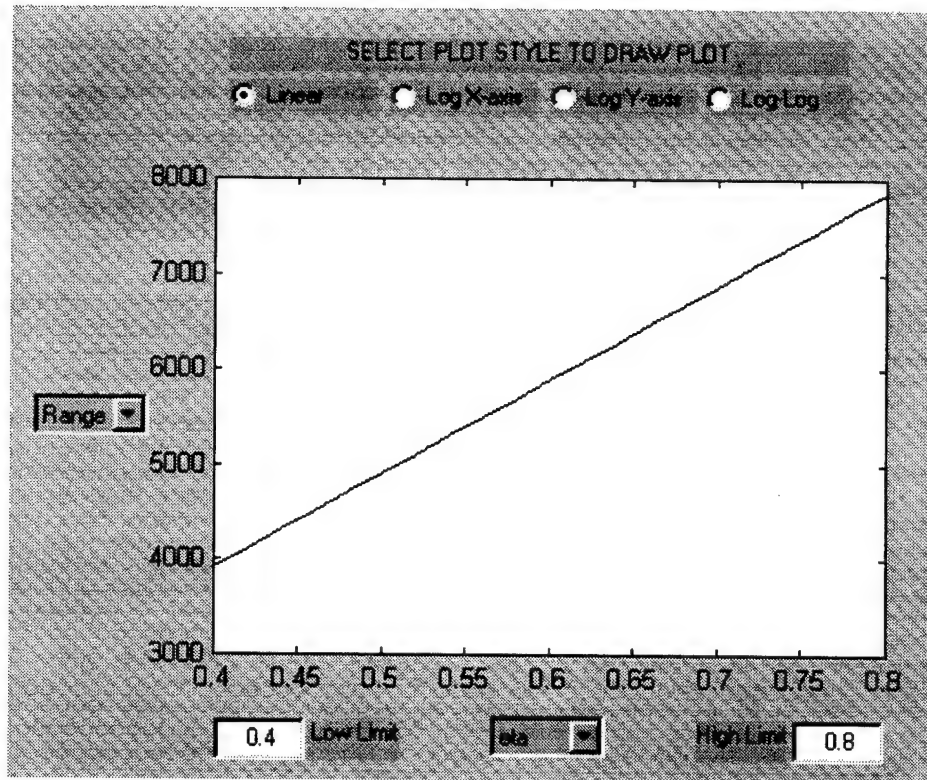
Figure 20. Range vs. Eta
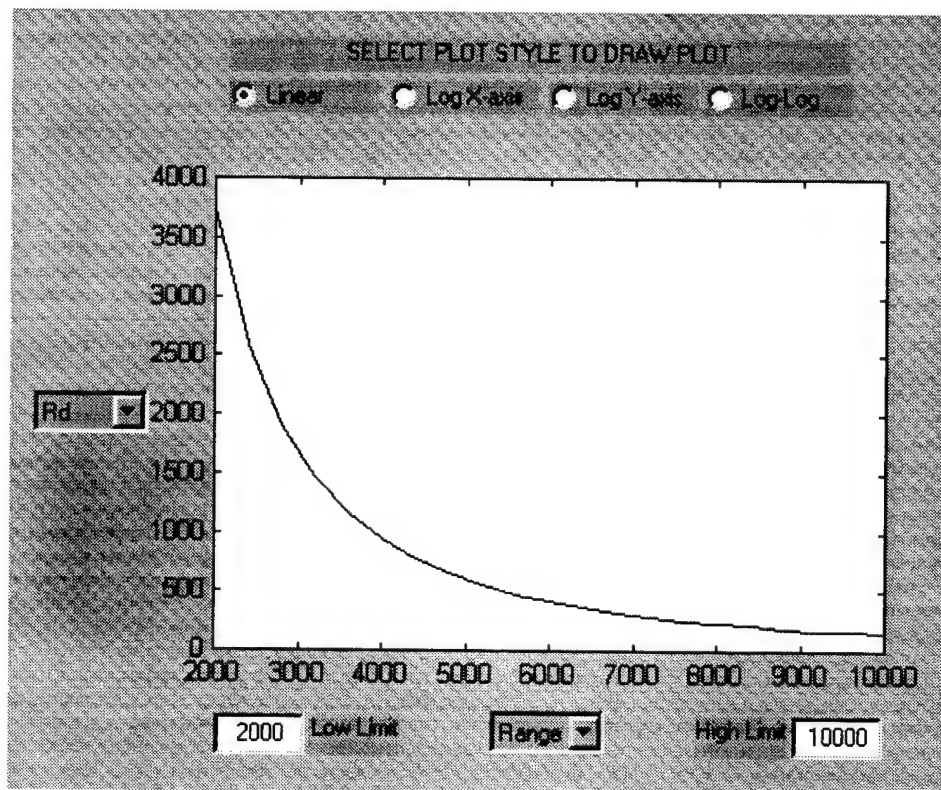
Figure 21. Rate vs. Range
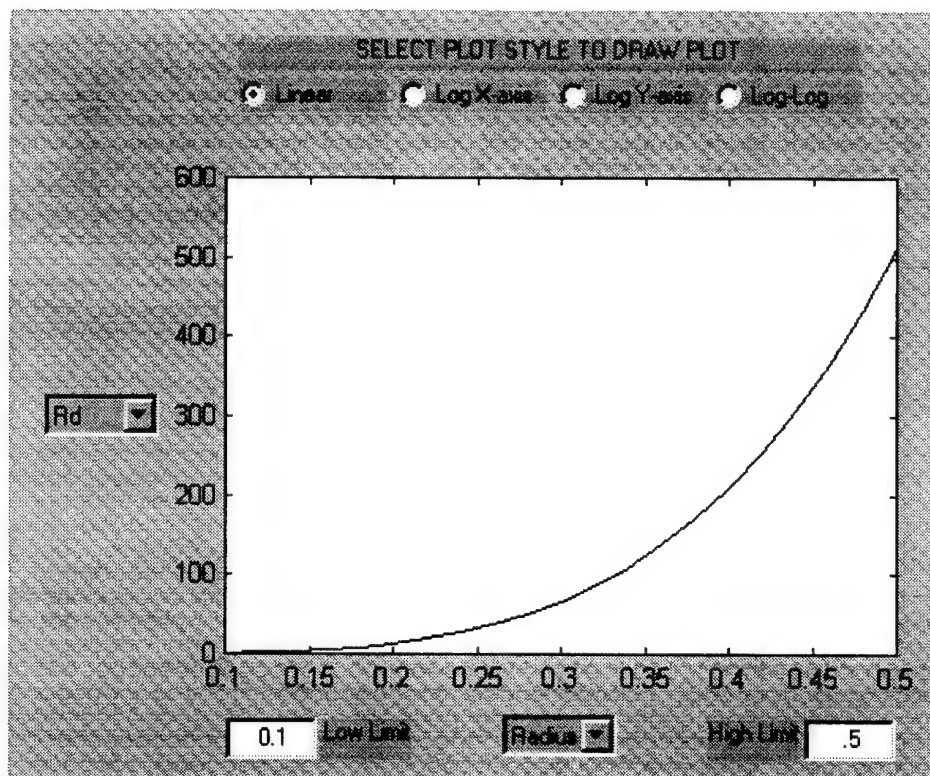
Figure 22. Rate vs. Radius
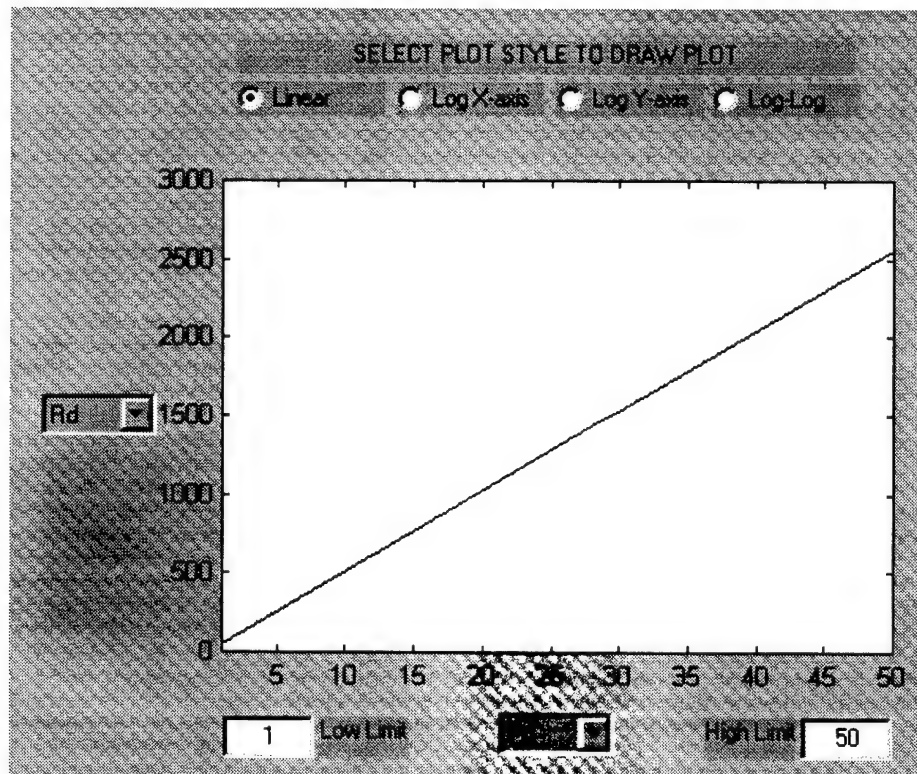
Figure 23. Rate vs. Power
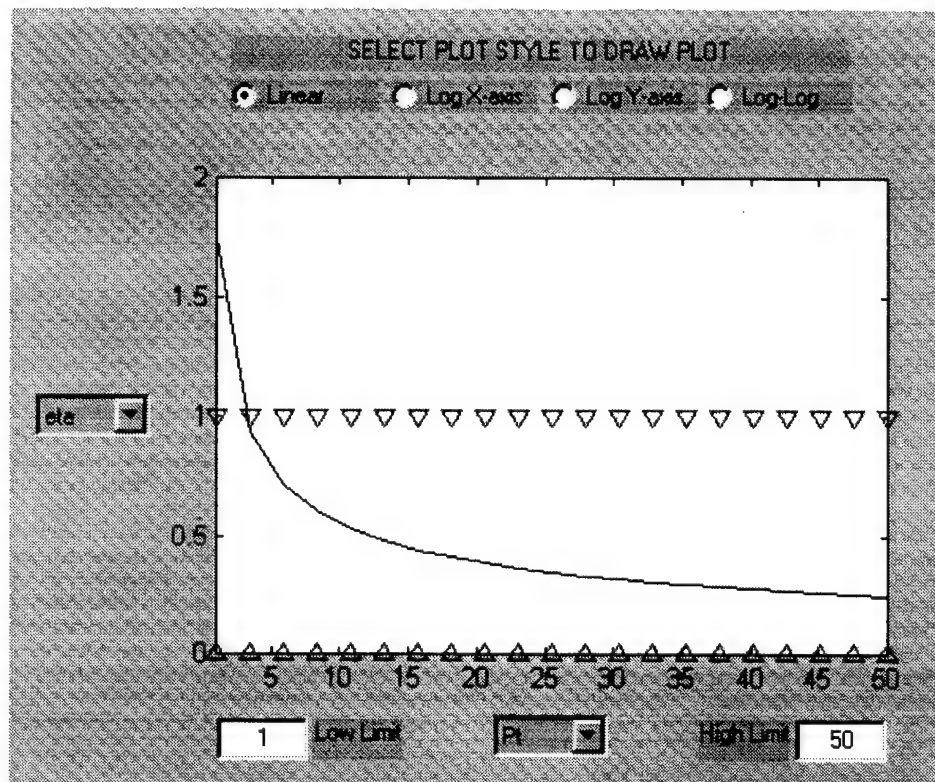
Figure 24. Eta vs. Power
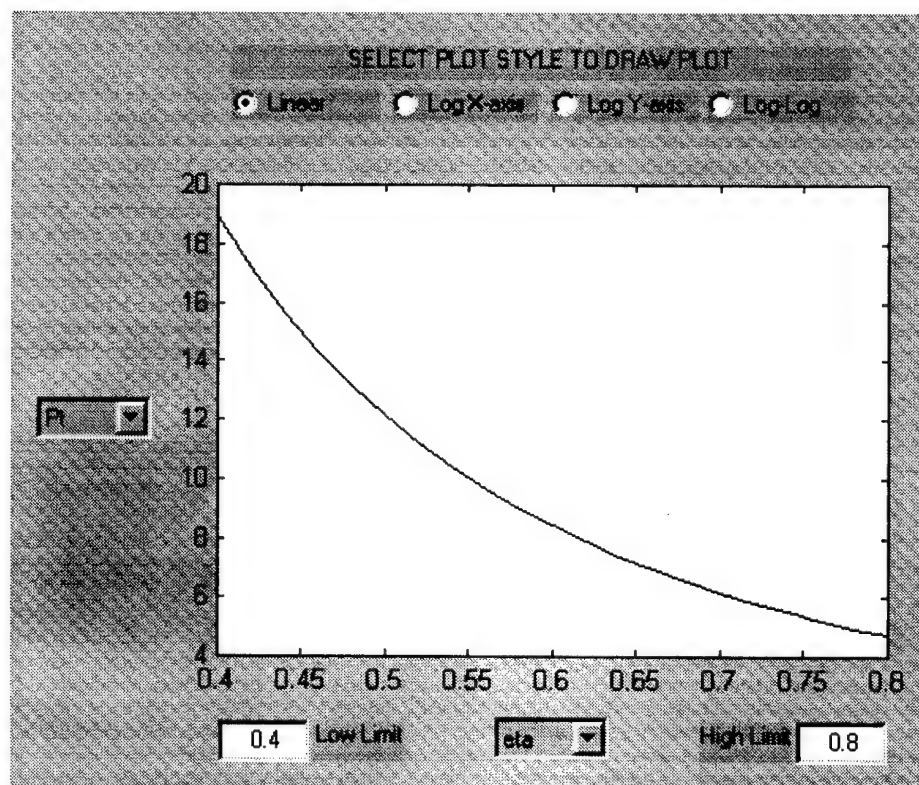
Figure 25. Power vs. Eta

61

trade-off between Range and Radius, Power, and eta. Figures 21 through 23 plot the trade-off between Data Rate and Range, Radius, and Power. Figure 24 plots the trade-off between eta and Power. Figure 25 plots the trade-off between Power and eta. Figures 16 and 17 show that Range increases with Radius. Figure 16 is a linear style plot whereas Figure 17 is a log-log style plot. Note that Figures 16 and 17 have different high Radius limits. The slope of Figure 17 is greater than one (i.e., for each decade increase in Radius, there is more than one-decade increase in Range). This indicates the significant impact of Radius upon Range. Figures 18 and 19 show that Range increases with Power. Figure 18 is a linear style plot while Figure 19 is a log-log style plot. The slope of Figure 19 is less than one. This indicates that, for this example, Power does not have as significant an affect upon Range as Radius does. Figure 20 shows that Range increase in a linear manner as eta increases. Figure 21 shows that Rate (the maximum possible data rate) decreases as Range increases. If viewed as a log-log style plot, the Rate vs. Range trade-off is a straight line with slope less than negative one. Figure 22 shows that Rate increases with Radius. Radius has a very significant affect on Rate; if viewed as a log-log style plot, each decade increase in Radius causes approximately a four-decade increase in Rate. Figures 16, 17 and 22 indicate that Radius is a very important factor in an ISL. Figure 23 shows that Rate increases in a linear manner as Power increases. Figures 24 and 25 show the trade-off between the same two parameters from different perspectives. Figure 24 shows that, in this example, the link cannot be established if Power is less than three watts because that would require eta to be greater than one. With all other parameters in this example held constant, for Power between three and approximately twelve watts, the required antenna efficiency decreases from 1 to 0.5. As

Power increases above twelve watts, the required antenna efficiency continues to decrease, but not nearly as quickly. Figure 25 shows that the required Power decreases as eta increases.

4.5   Summary

The Satellite Range Calculations GUI presents limiting cases for an ISL when the minimum LOS height enters the atmosphere or grazes the earth's surface. Range is an important factor in communication systems because free space loss, which is the largest loss factor in the link equation, is a function of range. The RF Tradeoffs GUI presents limiting cases for a communications system when the calculated value of the parameter of interest is not a valid value for the parameter. Significant parameters in the link equation can be analyzed with the RF Tradeoffs GUI. This is accomplished by analyzing the change in a selected parameter as the parameter of interest is varied. This is not a full-proof analysis because the main parameters have different units. The RF Graphs GUI provides powerful analysis of trade-offs because the graphs show trends over a range of data. With a trade-off selected, a reference parameter can be varied to show the trade-off with respect to the value of the reference parameter. Certain parameters in the link equation are more important than others. Range is significant, as discussed above. Radius and Power are significant considerations in satellite design. Eta is significant because satellite design considerations can affect it. Rate is significant because it is usually specified as a system requirement.

5 Discussion and Recommendations

5.1 Introduction

This thesis analyzes the communications capability of ISLs in a constellation of low earth orbiting satellites. The following sections summarize the work involved in the analysis, the significant findings, discuss considerations of RF and lasercom ISLs, and discuss areas of improvement and follow-on work.

5.2 Review of Thesis

The first step in this thesis analyzed orbital mechanics to determine the range between satellites in an ISL. As mentioned previously, range contributes to free space loss. Free space loss is the largest single loss factor in the link equation. The Satellite Range Calculations GUI computed the range between satellites for a given set of satellite constellation parameters. The Satellite Range Calculations GUI used several simplifying assumptions in its calculations. Possible improvements to the Satellite Range Calculations GUI are discussed in Section 5.5.

The thesis analyzed the components of the RF link equation. The analysis started with basic forms of the link equation (Equations 7 and 22) and developed a more detailed version of the link equation (Equation 47). Equation 47 allowed for analysis of the following parameters: transmitter power, the required ratio of energy per bit to noise energy density (E_b/N_o, includes coding), data rate, link margin, antenna line and coupling losses, antenna radius, antenna efficiency, frequency, range, satellite and antenna pointing loss, and effective system noise temperature. The RF Tradeoffs GUI was used to analyze these parameters; it solved for any unknown parameter when given the other

parameters. The RF Tradeoffs GUI required detailed derivation of the mathematical relationship between the probability of bit error, coding gain, modulation scheme, and E_b/N_o. The link equation used E_b/N_o as a parameter; however, a typical communications system design specifies a value for the probability of bit error. The value of E_b/N_o used in the link equation was calculated from the values of the probability of bit error and coding gain, and the desired modulation scheme. The RF Graphs GUI extended the analysis capability of the RF Tradeoffs GUI. The graphs visually show the trade-off between two parameters over a range of data. Both the RF Tradeoffs GUI and the RF Graphs GUI provided basic error checking on the parameter of interest. The GUIs indicated when the calculated value of the parameter of interest was physically invalid.

Analysis of the link equation indicated that antenna radius is a very significant factor in the link equation. The antenna radius has large effects on the trade-offs with both range and data rate. In addition, the antenna radius is an important satellite design consideration because of the antenna's weight and size.

5.3   Radio Frequency Communications ISL Considerations

Early background research for this thesis found a general guideline stating that at data rates greater than approximately 10 Mbps, a lasercom ISL may be advantageous over a RF ISL. Since this thesis used 512 Mbps as a default data rate, the general guideline indicated that RF ISLs would not be very practical. However, the general guideline was based upon ISLs for satellites in high-altitude orbits. Satellites in geostationary orbit are at an altitude of 35864 km. For eight satellites in geostationary orbit, spaced 45 degrees apart, the ISL range between the satellites is 32331 km. This range value is much greater than ranges encountered in the average constellation of low-

earth orbiting satellites. Table 10 compares the altitudes, ISL ranges and free space losses, and maximum data rates of a geostationary and low-earth orbiting satellite system.

Table 10. Comparison of Geostationary and LEO Satellite Systems

Type of System	Altitude	ISL Range	ISL Free Space Loss	Max R_D
Geostationary	35864	32331	-218.2	3.94
Low-Earth	700	5417	-202.68	140.18
Notes: 1. Altitude and ISL Range in km; ISL Free Space Loss in dB; Max R_D in Mbps. 2. ISL Range is based on In-Plane Range with 8 satellites per plane. 3. Other parameters: E_b/N_o: 4.4 dB; Margin: 2 dB; frequency: 60 GHz; P_t: 10 W; $\quad$ L_l: 1 dB; eta: 0.55; SNT: 34.87 dB-K; L_a: 3 dB; Radius: 0.2 m				

In Table 10, Radius is selected as 0.2 m, which is approximately the same size as the telescope arrays of lasercom systems [Kor97]. The orbital plane of geostationary satellites is the equatorial plane. The Satellite Range Calculations GUI can calculate ranges for geostationary satellites by considering them to be in the same orbital plane and ignoring the number of orbital planes and adjacent plane information. Table 10 shows that the shorter ISL range of low-earth orbiting satellites means that much higher data rates are possible. Thus RF ISLs are feasible for constellations of low-earth orbiting satellites. This finding caused this thesis research to focus on detailed analysis of the RF link equation.

A consideration that may limit the practicality of RF ISLs is spectrum allocation and regulation. As mentioned in Section 2.3, the 60 GHz band is specifically designated for use with ISLs. There are a limited number of frequency channels available for use in this band and as more ISLs are brought on-line, channel availability and interference will become important issues.

5.4 Lasercom ISL Considerations

As discussed in Section 2.4, lasercom ISL technology is not as mature as RF ISL technology. The background research for this these was not sufficient to develop a lasercom GUI. The overall difficulty encountered in the research was that a useful form of the laser link equation (Equation 28) was not found. The lasercom research did not succeed in developing Equation 28 into a more detailed link equation, similar in form to the RF link equation (Equation 47). A specific difficulty encountered during the lasercom background research involved transmitter telescope gain. Older lasercom designs tend to use large telescopes with Cassegrain optics. Cassegrain optics have a secondary mirror which obscures (blocks) a portion of the light entering/exiting the telescope. The telescope gain equation for Cassegrain optics is well developed in the literature [KlD74, DeK74]. As noted in Section 2.4.4, newer lasercom designs employ multiple transmit lasers (at the same wavelength) where each laser has its own telescope. The individual telescopes are usually small and unobstructed (not Cassegrain optics). An unobstructed telescope uses a different gain equation than a Cassegrain telescope. The beam combining effects of the multiple telescopes requires additional mathematics [Bis98]. Thus the overall transmitter telescope gain depends on several parameters.

Development of a lasercom GUI is an obvious area for follow-on research. More detailed research on the laser link equation would have to be done. The lasercom GUI would have to account for the variations available in lasercom technology, such as the multiple transmit lasers just discussed. This could be done by developing a GUI that uses parameters such as the type, size, and number of telescopes to calculate the transmit gain. With sufficient research, a lasercom GUI can probably be implemented in a form similar

to the RF GUIs where the value of a parameter of interest can be calculated or a trade-off between two parameters can be graphed.

5.5 GUI Improvements

All three GUIs in this thesis have the characteristic that they return errors if alphabetic characters or an empty string is entered where only numeric input is expected. A possible improvement to the GUIs is to improve error handling in this regard. If characters are entered, the GUI should cease processing and indicate to the user where the input needs to be corrected. The structure of the existing callback code may make implementation of this error handling difficult. The callback code nests the command to retrieve the input inside a command performing an operation on that input. The error is caused by the 'outside' command performing an operation on the input. In the following line of code, the "get" command is nested inside the "eval" command:

$$EbNo=eval(get(show\_ebno,'String'));$$

The "get" command retrieves the input. It is not sensitive to the input type (numeric, alphabetic, empty). The "eval" command operates on the input. It is sensitive to the input type and will return an error for alphabetic or empty input. A possible implementation of the error handling would require un-nesting the commands so that the input is retrieved, then the input is checked for error, and then (if not in error) execute the command performing an operation on the input.

All three GUIs indicate when the output calculated by the GUI is not a valid value. The Satellite Range Calculations GUI displays red text when the LOS passes through the atmosphere or grazes the earth's surface. The RF Tradeoffs GUI displays the parameter of interest with red text when its calculated value is invalid. The RF Graphs GUI uses

68

maximum and minimum valid limit indicators on its plots. The GUIs do not perform error checking on the parameters that are directly entered by the user. For example, negative values can be directly entered into offset angle edit box on the Satellite Range Calculations GUI. As noted in Table 1, for negative values of offset angle, the GUI will calculate physically invalid results but not return an error. A possible improvement to the GUIs is to improve error handling in this regard. After an input is retrieved, it should be checked against a table of valid values. If the input is invalid, the GUI should cease processing and indicate to the user where the input needs to be corrected. This error handling would be easier to implement than the alphabetic/empty string error handling discussed above.

The Satellite Range Calculations GUI assumes the satellites are in circular polar orbits. This assumption simplifies the calculations but is not very practical. This GUI (and the RF Tradeoffs GUI) attempted to make up for this weakness by allowing the user to directly enter a range value. A possible improvement to the Satellite Range Calculations GUI would increase its accuracy and flexibility. The GUI should be able to calculate the ranges for satellites in inclined orbits and should be able to use other well-known satellite constellations (e.g., Walker constellations).

The RF Tradeoffs GUI has four pre-set modulation schemes (BPSK, QPSK, 8-ary PSK, 16-ary PSK). A possible improvement to this GUI would increase the number of available modulation schemes (e.g., quadrature amplitude modulation schemes).

The plotting features of the RF Graphs GUI can be improved. A 'multiple plot' feature that allows overlaying of multiple plots is an improvement, especially when plotting parametric trade-offs, as discussed in Section 4.4.1. Another improvement

69

would display the coordinates of the mouse pointer. The coordinates could be displayed either in edit boxes located near the plot or displayed next to the pointer when the right mouse button is clicked. Other improvements are automated 'zoom' and 'pan' feature that would be operated by clicking or dragging the mouse over the plot.

5.6 Summary

This thesis analyzes the communications capability of RF ISLs. While it is aimed at analyzing the ISLs of a constellation of low earth orbiting satellites, it is general enough to analyze free-space ISLs with any arbitrary range at any operating frequency. The GUIs perform calculations on the parameters entered by the user. The Satellite Range Calculations GUI calculates the range between satellites. The RF Tradeoffs GUI can solve for any parameter of interest when given values for the other parameters. The RF Graphs GUI visually plots the trade-off between two parameters.

Appendix A: Index of Code

The GUIs built with Matlab's® Guide tools for this thesis consist of three separate files. The first file is the actual GUI; it contains the information about the size, location, and properties of the components of the GUI. This is an m-file (the file extension is ".m") and it can be viewed and edited with Matlab's® editor. However, Matlab® 'hides' some of the details of the code and simply displays ellipses, as shown in the code below:

```
h0 = figure('Units','inches', ...
            'Color',[0.8 0.8 0.8], ...
            'Colormap',mat0, ...
            'CreateFcn','comm_graph_code initial', ...
```

The second file is a binary data file used by the GUI. It is a mat-file (the file extension is ".mat") with the same filename as the GUI. It cannot be viewed with Matlab's® editor. The third file contains the callback code modules written by the GUI designer. This is an m-file. Writing the callback code modules in a separate file is called 'switchyard programming' and is described in "Building GUIs with MATLAB®." MATLAB® documentation (Version 5), The MathWorks, Inc., 1997, chapter-page 3-27.

The three GUIs are listed below. Under each GUI name is a list of the individual files (and associated information) which make up the GUI. The associated information is the size and save date of the file, the version number, and descriptive notes. Version numbers apply to the GUI m-files and the callback code m-files; the version number is given in the comment lines at the beginning of the file. The comment lines in the files contain a "Last Update" date which corresponds to the save date of the file.

Satellite Range Calculations GUI:

Filename	Size	Date	Version	Notes
sat_range_gui.m	10 KB	2/26/99	V1.0	The GUI file.
sat_range_gui.mat	5 KB	2/25/99	n/a	Mat file.
sat_range_code.m	7 KB	2/26/99	V1.2	Callback code. See

Appendix B for code (page 73).

RF Tradeoffs GUI:

Filename	Size	Date	Version	Notes
comm_sel_gui3.m	23 KB	2/28/99	V1.0	The GUI file.
comm_sel_gui3.mat	5 KB	1/10/99	n/a	Mat file.
comm_sel_code3.m	29 KB	2/26/99	V5.2	Callback code. See

Appendix C for code (page 77).

RF Graphs GUI:

Filename	Size	Date	Version	Notes
comm_graph_gui.m	17 KB	2/28/99	V1.0	The GUI file.
comm_graph_gui.mat	9 KB	2/15/99	n/a	Mat file.
comm_graph_code.m	42 KB	2/27/99	V6.3	Callback code. See

Appendix D for code (page 90).

For electronic copies of the files, send an email request to:
afeltman1990@alum.mit.edu
richard.raines@afit.af.mil

Appendix B: Satellite Range Calculations GUI

sat_range_code.m

```
function sat_range_code(action)
% SAT_RANGE_CODE.M   This file supports the pushbutton operations of GUI
% named sat_range_gui.m  Enter "help sat_range_gui" to get more information.
%
% SAT_RANGE_CODE.M is not a stand-alone m-file.
%
% Capt. Andy Feltman
% V1.2; Last Update: 26 Feb 99 (Matlab V5.2 PC)

%Very important!!  Declare global variables before switches
%Global variables can be passed between case statements and between GUIs
global IN_PLANE_RANGE ADJ_PLANE_RANGE %initialized in sat_range_code

%Do all findobj commands once! (gets the handles for get and set commands)
show_spp=findobj(gcbf,'Tag','show_spp');
show_np=findobj(gcbf,'Tag','show_np');
show_ang=findobj(gcbf,'Tag','show_ang');
show_alt=findobj(gcbf,'Tag','show_alt');
show_hoa=findobj(gcbf,'Tag','show_hoa');
show_ipr=findobj(gcbf,'Tag','ip_range');
show_apr=findobj(gcbf,'Tag','ap_range');
show_iph=findobj(gcbf,'Tag','ip_hoa');
show_aph=findobj(gcbf,'Tag','ap_hoa');

switch(action)

case 'sl1', %slider action on satellites per plane
   t1=round(get(gcbo,'Value')); %retrives the slider's value
   %used gcbo to self-reference to current slider
   set(show_spp,'String',t1) %set sat-per-plane edit box string to slider value

case 'sl2', %slider action on number of planes
   t2=round(get(gcbo,'Value'));
   set(show_np,'String',t2)

case 'sl3', %slider action on altitude
   t3=round(get(gcbo,'Value'));
   set(show_alt,'String',t3)

case 'sl4', %slider action on height of atmosphere
   t4=round(get(gcbo,'Value'));
   set(show_hoa,'String',t4)
```

```matlab
case 'sl5', %slider action on angular offset between planes
   t5=round(get(gcbo,'Value'));
   set(show_ang,'String',t5)

case 'calc', %action on calculate ranges and heights button
   %Get variables
   nspp=eval(get(show_spp,'String')); %number of satellites per plane
   np=eval(get(show_np,'String')); %number of orbital planes
   angoff=eval(get(show_ang,'String')); %offset angle (degrees)
   alt=eval(get(show_alt,'String')); %satellite altitude
   hoa=eval(get(show_hoa,'String')); %height of atmosphere

   %Compute ranges and heights
   %***********************
   %Initialize variables
   re=6378; %earth radius
   atm_leg=re+hoa; %center of earth to top of atmosphere
   sat_leg=re+alt; %center of earth to satellite
   %In/adjacent plane distance between satellites (angles measured in radians)
   ip_ca=(2*pi)/nspp; %in plane central angle
   ap_ca=pi/np; %adj plane central angle (factors of 2 cancel in numerator/denominator)
   ip_sa=(pi-ip_ca)/2; %in plane satellite angle
   ap_sa=(pi-ap_ca)/2; %adj plane satellite angle
   %in plane range calc (law of sines)
   in_p_r=(sat_leg*sin(ip_ca)/sin(ip_sa));
   %adj plane range calc (measured at equator)
   temp_adj_p_r=(sat_leg*sin(ap_ca)/sin(ap_sa)); %straight distance between planes
   adj_p_r=temp_adj_p_r/cos(angoff*pi/180); %distance include angle offset
   %Min altitude of in plane cross-link beam (simplify law of sines)
   min_height_ip=sat_leg*sin(ip_sa);
   min_h_ip=min_height_ip-re; %relative to earth
   %Min altitude of adj plane cross-link beam
   %recompute angles due to offset angle
   offset_cent_ang=asin(adj_p_r/(2*sat_leg)); %radians!
   offset_sat_ang=pi/2-offset_cent_ang; %radians!
   min_height_ap=sat_leg*sin(offset_sat_ang);
   min_h_ap=min_height_ap-re; %relative to earth

   %Display ranges and heights
   set(show_ipr,'String',in_p_r); %disp in plane range
   set(show_apr,'String',adj_p_r); %disp adj plane range
   set(show_iph,'String',min_h_ip); %disp in plane min LOS height
   if min_h_ip<hoa %if in plane min LOS height enters atmosphere, show red text
      set(show_iph,'ForegroundColor',[1 0 0]);
   else set(show_iph,'ForegroundColor',[0 0 0]);
```

```
        end
        set(show_aph,'String',min_h_ap); %disp adj plane min LOS height
        if min_h_ap<hoa %if adj plane min LOS height enters atmosphere, show red text
            set(show_aph,'ForegroundColor',[1 0 0]);
        else set(show_aph,'ForegroundColor',[0 0 0]);
        end

    case 'plot_orbit', %plots one orbital plane
        %Recompute ranges and heights
        sat_range_code calc

        %Values computed in 'calc' must be re-calculated for use here
        %(non-global variables can't pass between calls!)
        nspp=eval(get(show_spp,'String')); %number of satellites per plane
        alt=eval(get(show_alt,'String')); %satellite altitude
        hoa=eval(get(show_hoa,'String')); %height of atmosphere
        re=6378; %earth radius
        atm_leg=re+hoa;
        sat_leg=re+alt;
        ip_ca=(2*pi)/nspp; %in plane central angle

        %Setup for plotting & calculations
        %POLAR plot command format: polar(angle,magnitude)
        %angle and magnitude must be same length vectors

        %1x361 vector of 360 degrees converted to radians; used as angle in polar command
        ang=[0:1:360]*pi/180;
        %1x361 vector used to 'vectorize' magnitude in polar command
        circs=ones(1,length(ang));
        %vector of angular location of satellites; +pi/2 puts satellite at north pole
        sat_angs=ip_ca*[0:1:nspp-1]+pi/2; %length of vector equals # of satellites per plane
        %vector same length as sat_angs used to 'vectorize' magnitude in polar command
        scirc=ones(1,length(sat_angs));
        %vector of angluar location of direct LOS lines
        los=[sat_angs pi/2]; %same as sat_angs but 'completes circle' at north pole
        %vector same length as los used to 'vectorize' magnitude in polar command
        los_len=[scirc 1];
        %End setup

        %PLOTTING ROUTINE
        figure;clf;hold on;axis equal;
        %ang and circs are 1x361
        polar(ang,re*1.5*circs,'w'); %plots extra white space border for LEOs
        polar(ang,re*circs,'b'); %plots earth outline
        polar(ang,atm_leg*circs,'g'); %plot atmosphere
        polar(ang,sat_leg*circs,'k:'); %plots sat orbit outline as dashed line
```

```
%sat_angs and scirc have length equal to number of satellites per orbital plane
polar(sat_angs,sat_leg*scirc,'k+'); %plots satellite positions
%los and los_len have length equal to (# of satellites per plane) + 1
polar(los,sat_leg*los_len,'r'); %plots direct LOS lines
compass(-pi/2,re);text(-200,6650,'N'); %North pole arrow
np=eval(get(show_np,'String')); %number of orbital planes needed in title line
title(sprintf('Position of Satellites in 1 of %d Orbital Planes',np));
xlabel('True Relative Earth Size/Satellite Altitude');
ylabel('Distance from Center of Earth (km)');
set(gca,'Xtick',[]); %Eliminate x-axis marks
set(gca,'Ytick',[-sat_leg -re 0 re sat_leg]); %Change y-axis marks
hold off; %END PLOTTING ROUTINE

case 'rf', %Goes to RF crosslink calculation page
   %set values of global variables to pass to RF Tradeoffs GUI
   IN_PLANE_RANGE=eval(get(show_ipr,'String')); %set in-plane range
   ADJ_PLANE_RANGE=eval(get(show_apr,'String')); %set adj-plane range
   comm_sel_gui3 %launch RF Tradeoffs GUI

end %end switch
```

Appendix C: RF Tradeoffs GUI

comm_sel_code3.m

```
function comm_sel_code3(action)
% COMM_SEL_CODE3.M   This file supports the pushbutton operations of GUI
% named comm_sel_gui3.m  Enter "help comm_sel_gui3" to get more information.
%
% COMM_SEL_CODE3.M is not a stand-alone m-file.
%
% Capt. Andy Feltman
% V5.2; Last Update: 26 Feb 99 (Matlab V5.2 PC)


%Very important!!  Declare global variables before switches
global IN_PLANE_RANGE ADJ_PLANE_RANGE %imported from range_code
(ranges in km)
%the following globals initialized in comm_sel_code3
global range EbNo rd rddb marg lamb PtW Pt Ll eta snt La rad %basic parameters
global fsl gain beamwidth gt %system parameters
%********************
fgc='ForegroundColor'; %cleans up 'calc_xxxx' case statements (and others)


%Do all findobj commands once! (gets the handles!)
show_ipr=findobj(gcbf,'Tag','show_ipr');
ipr_butt=findobj(gcbf,'Tag','ipr_butt');
show_apr=findobj(gcbf,'Tag','show_apr');
apr_butt=findobj(gcbf,'Tag','apr_butt');
sel_range=findobj(gcbf,'Tag','sel_range'); %the range parameter radiobutton
show_range=findobj(gcbf,'Tag','show_range'); %the range parameter editbox
sel_ebno=findobj(gcbf,'Tag','sel_ebno');
show_ebno=findobj(gcbf,'Tag','show_ebno');
sel_rd=findobj(gcbf,'Tag','sel_rd');
show_rd=findobj(gcbf,'Tag','show_rd');
sel_m=findobj(gcbf,'Tag','sel_m');
show_m=findobj(gcbf,'Tag','show_m');
sel_wave=findobj(gcbf,'Tag','sel_wave');
show_wave=findobj(gcbf,'Tag','show_wave');
show_freq=findobj(gcbf,'Tag','show_freq'); %no radiobutton for frequency
sel_pt=findobj(gcbf,'Tag','sel_pt');
show_pt=findobj(gcbf,'Tag','show_pt');
sel_ll=findobj(gcbf,'Tag','sel_ll');
show_ll=findobj(gcbf,'Tag','show_ll');
sel_eta=findobj(gcbf,'Tag','sel_eta');
show_eta=findobj(gcbf,'Tag','show_eta');
sel_snt=findobj(gcbf,'Tag','sel_snt');
show_snt=findobj(gcbf,'Tag','show_snt');
```

```
show_to=findobj(gcbf,'Tag','show_to'); %no radiobuttons for To, Ta, Tr, L
show_ta=findobj(gcbf,'Tag','show_ta');
show_tr=findobj(gcbf,'Tag','show_tr');
show_l=findobj(gcbf,'Tag','show_l');
sel_point=findobj(gcbf,'Tag','sel_point'); %'point' refers to La
show_point=findobj(gcbf,'Tag','show_point');
show_track=findobj(gcbf,'Tag','show_track'); %no radiobutton for tracking accuracy
sel_rad=findobj(gcbf,'Tag','sel_rad');
show_rad=findobj(gcbf,'Tag','show_rad');
show_parameter=findobj(gcbf,'Tag','show_parameter'); %indicates the unknown
parameter
show_open=findobj(gcbf,'Tag','show_open'); %the editbox showing the value of the unk
param
show_fsl=findobj(gcbf,'Tag','show_fsl');
show_antgain=findobj(gcbf,'Tag','show_antgain');
show_3dbbw=findobj(gcbf,'Tag','show_3dbbw');
show_gtfom=findobj(gcbf,'Tag','show_gtfom');
do_graph=findobj(gcbf,'Tag','graph_it'); %graph button
PB_desired=findobj(gcbf,'Tag','PB_desired'); %Pb edit box
bpsk=findobj(gcbf,'Tag','bpsk'); %bpsk,qpsk,8-ary,16-ary modulation scheme
radiobuttons
qpsk=findobj(gcbf,'Tag','qpsk');
opsk=findobj(gcbf,'Tag','opsk');
hpsk=findobj(gcbf,'Tag','hpsk');
c_gain=findobj(gcbf,'Tag','c_gain'); %coding gain edit box

switch(action)

case 'initial', %action on opening GUI (called by CreateFcn of show_ipr)
    %display in-plane range and adjacent-plane range in appropriate edit boxes
    set(show_ipr,'String',IN_PLANE_RANGE);
    set(show_apr,'String',ADJ_PLANE_RANGE);

case 'range', %action on selecting in-plane or adj-plane range
    set(show_ipr,'String',IN_PLANE_RANGE); %re-display in_p_r; overrides direct entry
    set(show_apr,'String',ADJ_PLANE_RANGE); %re-display adj_p_r
    %Button check routine sets range that will be passed to calculation
    tempuse=get(gcbo,'Tag'); %find which radiobutton was selected
    if tempuse=='ipr_butt' %if ipr selected... (3 things)
        set(gcbo,'Value',1); %1. turn on ipr radiobutton
        set(apr_butt,'Value',0); %2. turn off apr radiobutton
        set(show_range,'String',IN_PLANE_RANGE); %3. show ipr in range parameter edit
box
    elseif tempuse=='apr_butt' %if apr selected... (3 things)
        set(gcbo,'Value',1); %1. turn on apr radiobutton
        set(ipr_butt,'Value',0); %2. turn off ipr radiobutton
```

```
            set(show_range,'String',ADJ_PLANE_RANGE); %3. show apr in range parameter
edit box
    end

case 'calc_range',
    %Turn on appropriate button, set its edit box text to white
    %Turn off all other buttons, set their edit boxes text to black
    %"fgc" is abbreviation for ForegroundColor
    set(sel_range,'Value',1);set(show_range,fgc,[1 1 1]); %white text in range edit box
    set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]); %black text in ebno edit box
    set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]); %black text for Pb and coding gain
    set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
    set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
    set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
    set(show_freq,fgc,[0 0 0]); %black text for frequency
    set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
    set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
    set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
    set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
    set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]); %black text for To, Ta, Tr, L
    set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
    set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
    set(show_track,fgc,[0 0 0]); %black text for tracking accuracy
    set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
    set(show_parameter,'String','Range '); %show "Range" in 'solve for' box
    %white text in compute box--'clears' previous value; reset black text when computing
    set(show_open,fgc,[1 1 1]);

case 'calc_ebno',
    set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
    set(sel_ebno,'Value',1); set(show_ebno,fgc,[1 1 1]); %white text in ebno edit box
    set(PB_desired,fgc,[1 1 1]);set(c_gain,fgc,[1 1 1]); %white text for Pb and coding gain
    set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
    set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
    set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
    set(show_freq,fgc,[0 0 0]);
    set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
    set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
    set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
    set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
    set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
    set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
    set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
    set(show_track,fgc,[0 0 0]);
    set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
    set(show_parameter,'String','Eb/No ');
```

```
            set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_rd',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',1);set(show_rd,fgc,[1 1 1]); %white text in data rate edit box
      set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String',' Rd ');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_m',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
      set(sel_m,'Value',1); set(show_m,fgc,[1 1 1]); %white text in margin edit box
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String','Margin');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_wave',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
```

```
    set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
    set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
    set(sel_wave,'Value',1);set(show_wave,fgc,[1 1 1]); %white text in lambda edit box
    set(show_freq,fgc,[1 1 1]); %white text in frequency edit box
    set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
    set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
    set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
    set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
    set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
    set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
    set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
    set(show_track,fgc,[0 0 0]);
    set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
    set(show_parameter,'String','Lambda');
    set(show_open,fgc,[1 1 1]); %white out compute box

case 'calc_pt',
    set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
    set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
    set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
    set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
    set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
    set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
    set(show_freq,fgc,[0 0 0]);
    set(sel_pt,'Value',1);set(show_pt,fgc,[1 1 1]); %white text in Pt edit box
    set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
    set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
    set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
    set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
    set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
    set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
    set(show_track,fgc,[0 0 0]);
    set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
    set(show_parameter,'String',' Pt ');
    set(show_open,fgc,[1 1 1]); %white out compute box

case 'calc_ll',
    set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
    set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
    set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
    set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
    set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
    set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
    set(show_freq,fgc,[0 0 0]);
    set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
    set(sel_ll,'Value',1);set(show_ll,fgc,[1 1 1]); %white text in Ll edit box
```

```matlab
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String',' Ll ');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_eta',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
      set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',1);set(show_eta,fgc,[1 1 1]); %white text in eta edit box
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String',' eta ');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_snt',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
      set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',1);set(show_snt,fgc,[1 1 1]); %white text in SNT edit box
      set(show_to,fgc,[1 1 1]);set(show_ta,fgc,[1 1 1]); %white text for Ta, To, Tr, L
      set(show_tr,fgc,[1 1 1]);set(show_l,fgc,[1 1 1]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
```

82

```matlab
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String',' SNT ');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_point',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
      set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',1);set(show_point,fgc,[1 1 1]); %white text in La edit box
      set(show_track,fgc,[1 1 1]); %white text for tracking accuracy
      set(sel_rad,'Value',0);set(show_rad,fgc,[0 0 0]);
      set(show_parameter,'String',' La ');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'calc_rad',
      set(sel_range,'Value',0);set(show_range,fgc,[0 0 0]);
      set(sel_ebno,'Value',0);set(show_ebno,fgc,[0 0 0]);
      set(PB_desired,fgc,[0 0 0]);set(c_gain,fgc,[0 0 0]);
      set(sel_rd,'Value',0);set(show_rd,fgc,[0 0 0]);
      set(sel_m,'Value',0);set(show_m,fgc,[0 0 0]);
      set(sel_wave,'Value',0);set(show_wave,fgc,[0 0 0]);
      set(show_freq,fgc,[0 0 0]);
      set(sel_pt,'Value',0);set(show_pt,fgc,[0 0 0]);
      set(sel_ll,'Value',0);set(show_ll,fgc,[0 0 0]);
      set(sel_eta,'Value',0);set(show_eta,fgc,[0 0 0]);
      set(sel_snt,'Value',0);set(show_snt,fgc,[0 0 0]);
      set(show_to,fgc,[0 0 0]);set(show_ta,fgc,[0 0 0]);
      set(show_tr,fgc,[0 0 0]);set(show_l,fgc,[0 0 0]);
      set(sel_point,'Value',0);set(show_point,fgc,[0 0 0]);
      set(show_track,fgc,[0 0 0]);
      set(sel_rad,'Value',1);set(show_rad,fgc,[1 1 1]); %white text in Radius edit box
      set(show_parameter,'String','Radius');
      set(show_open,fgc,[1 1 1]); %white out compute box

  case 'reset_all', %reset all parameters to default values
      comm_sel_code3 calc_rd %sets Rd as parameter to solve for
```

```
    set(show_range,'String',7000); %range in km
    set(show_rd,'String',512); %Rd in Mbps
    set(show_ebno,'String',4.4); %Eb/No in dB
    set(show_m,'String',2); %Margin in dB
    set(show_wave,'String',.005); %Lambda in meters
    set(show_freq,'String',60); %Frequency in GHz
    set(show_pt,'String',10); %Pt in watts
    set(show_ll,'String',1); %Ll in dB
    set(show_eta,'String',.55); %eta is dim-less
    set(show_point,'String',3); %La in dB
    set(show_track,'String',.5); %Track Accuracy is dim-less
    set(show_rad,'String',.5); %Radius in meters
    %Set BPSK as default modulation scheme; turn its radiobutton on, others off
    set(bpsk,'Value',1);set(qpsk,'Value',0);set(opsk,'Value',0);set(hpsk,'Value',0);
    set(c_gain,'String',5.2); %Coding gain in dB
    set(PB_desired,'String',1e-5); %Pb is error/bit
    comm_sel_code3 reset_temps %reset temperature parameters

case 'reset_temps', %reset system noise temp parameters
    set(show_l,'String',1); %L in dB
    set(show_ta,'String',10); %Ta, To, Tr in K
    set(show_to,'String',300);
    set(show_tr,'String',3000);
    set(show_snt,'String',34.87); %SNT in dB-K

case 'freq_wave', %action on entering frequency or wavelength
    %Determine whether freq or wavelength was entered and calc the other
    vary=eval(get(gcbo,'String')); %retrieve whatever was entered
    test=get(gcbo,'Tag'); %check which was entered
    if test=='show_freq' %if a freq (GHz) was entered
        freq=vary; %set frequency to what was entered
        lamb=.3/freq; %calculate wavelength (meters) = c/freq
    elseif test=='show_wave' %if a wavelength (meters) was entered
        lamb=vary; %set wavelength to what was entered
        freq=.3/lamb; %calculate frequency (GHz) = c/wavelength
    end
    %Display
    set(show_freq,'String',freq); %display frequency (GHz) in freq box
    set(show_wave,'String',lamb); %display wavelength (meters) in lambda box

case 'find_ebno', %action to compute EbNo given modulation, Pb, and coding gain
    %find desired Pb, modulation, and coding gain
    PB_d=eval(get(PB_desired,'String')); %desired Pb
    %need to determine which modulation radiobutton pushed and assign k and M
    tempuse=get(gcbo,'Tag'); %gcbo self-references to the current radiobutton
    if tempuse=='bpsk' %BPSK k=1 M=2
```

```matlab
      set(gcbo,'Value',1); %turn on BPSK button
      set(qpsk,'Value',0); %turn off QPSK button
      set(opsk,'Value',0); %turn off 8-ary PSK button
      set(hpsk,'Value',0); %turn off 16-ary PSK button
      k=1; M=2; %set parameters
    elseif tempuse=='qpsk' %QPSK k=2 M=4
      set(bpsk,'Value',0);
      set(gcbo,'Value',1);
      set(opsk,'Value',0);
      set(hpsk,'Value',0);
      k=2; M=4;
    elseif tempuse=='opsk' %8-ary PSK k=3 M=8
      set(bpsk,'Value',0);
      set(qpsk,'Value',0);
      set(gcbo,'Value',1);
      set(hpsk,'Value',0);
      k=3; M=8;
    elseif tempuse=='hpsk' %16-ary PSK k=4 M=16
      set(bpsk,'Value',0);
      set(qpsk,'Value',0);
      set(opsk,'Value',0);
      set(gcbo,'Value',1);
      k=4; M=16;
    end %determine modulation button pushed
    code_gain=eval(get(c_gain,'String')); %coding gain
    %determine uncoded EbNo
    %Pb formula same for BPSK & QPSK; different formula for M-ary PSK
    if (k==1) | (k==2)
      ebno_absolute=(erfinv(1-2*PB_d))^2;
      EN_reqd=10*log10(ebno_absolute);
    elseif (k==3) | (k==4)
      term1=1/(2*k);
      term2=sqrt(2)/sin(pi/M);
      term3=1-k*PB_d;
      ebno_abs=term1*((term2*erfinv(term3))^2);
      EN_reqd=10*log10(ebno_abs);
    end
    EN_coded=EN_reqd-code_gain; %subtract coding gain
    set(show_ebno,'String',EN_coded); %display

 case 'comp_snt', %action to compute effective sys noise temp from raw temps
    L=eval(get(show_l,'String')); %L in dB
    Labs=10^(L/10); %Convert L (db) to Labs (absolute)
    Ta=eval(get(show_ta,'String'));
    To=eval(get(show_to,'String'));
    Tr=eval(get(show_tr,'String'));
```

```
      snt=10*log10((Ta/Labs)+((Labs-1)*To/Labs)+Tr); %absolutes converted to dB
      set(show_snt,'String',snt); %display in snt box

   case 'calc_pl', %action to calc pointing loss
      %pointing loss is wavelength indep when expressed as fraction of 3dB beamwidth
      %Determine whether La or fraction was entered and calc the other
      vary=eval(get(gcbo,'String')); %retrieve whatever was entered
      test=get(gcbo,'Tag'); %check which was entered
      if test=='show_point' %if La entered
         La=vary; %set La to what was entered
         track_fract=sqrt(La/12); %compute TrackAcc fraction
      elseif test=='show_track' %if fraction entered
         track_fract=vary; %set TrackAcc to what was entered
         La=12*track_fract*track_fract; %compute La
      end
      %Display
      set(show_point,'String',La); %display La
      set(show_track,'String',track_fract); %display TrackAcc

   case 'check_valid', %action to look for valid values of directly entered parameters
      %Valid values known for Eb/No, Ll, La, eta, Margin, Radius
      %Reserved for future

   case 'main_calculate', %action to solve for unknown parameter
      %Get all variables--most variables are globals
      range=1000*eval(get(show_range,'String')); %convert displayed km range to m
      EbNo=eval(get(show_ebno,'String')); %EbNo in dB
      rd=1e6*eval(get(show_rd,'String')); %convert displayed Mbps data rate to bps
      rddb=10*log10(rd); %rddb in db-bps
      marg=eval(get(show_m,'String')); %Margin in dB
      lamb=eval(get(show_wave,'String')); %lambda in m
      PtW=eval(get(show_pt,'String')); %PtW in Watts
      Pt=10*log10(PtW); %convert Pt to dB
      Ll=eval(get(show_ll,'String')); %Ll in dB
      eta=eval(get(show_eta,'String')); %eta is dim-less
      snt=eval(get(show_snt,'String')); %snt in dB-K
      To=eval(get(show_snt,'String')); %To in K; need this in addition to SNT
      Ta=eval(get(show_snt,'String')); %Ta in K; need this in addition to SNT
      Tr=eval(get(show_snt,'String')); %Tr in K; need this in addition to SNT
      L=eval(get(show_snt,'String')); %L in dB; need this in addition to SNT
      Labs=10^(L/10); %Convert L (db) to Labs (absolute)
      La=eval(get(show_point,'String')); %La in dB
      rad=eval(get(show_rad,'String')); %antenna radius in m

      %black text for compute box--'calc_xxx' statements previously set white text
      set(show_open,fgc,[0 0 0]);
```

```
%Determine which button is on (value=1) and solve for it!
%note these computations will over-ride values from get statements above
if (get(sel_range,'Value'))==1 %if solve for range
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    rt=Pt-EbNo-rddb-marg-Ll+2*gain-La+228.6-snt; %solve link eq for range term
    range=lamb*(10^(rt/20))/(4*pi); %solve for range in m
    set(show_open,'String',(range/1000)); %display km
elseif (get(sel_ebno,'Value'))==1 %if solve for ebno
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    EbNo=Pt-rddb-marg-Ll+2*gain+fsl-La+228.6-snt; %solve link eq for EbNo in dB
    set(show_open,'String',EbNo); %display
elseif (get(sel_rd,'Value'))==1 %if solve for data rate
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    rddb=Pt-EbNo-marg-Ll+2*gain+fsl-La+228.6-snt; %solve link eq for rd in dB-bps
    rd=(10^(rddb/10)); %rd in bps
    set(show_open,'String',(rd/1e6)); %display Mbps
elseif (get(sel_m,'Value'))==1 %if solve for margin
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    marg=Pt-rddb-EbNo-Ll+2*gain+fsl-La+228.6-snt; %solve link eq for M in dB
    set(show_open,'String',marg); %display
elseif (get(sel_wave,'Value'))==1 %if solve for wavelength
    %REALIZE LAMBDA IS IN GAIN AND FSL TERMS
    %solve link equation for lambda term (lamterm=20log(lamb))
    lamterm=Pt-EbNo-rddb-marg-Ll+20*log10(4*(pi^2)*(rad^2)*eta)-
20*log10(4*pi*range)-La+228.6-snt;
    lamb=10^(lamterm/20); %convert lamb to m
    set(show_open,'String',lamb); %display only wavelength, not frequency
elseif (get(sel_pt,'Value'))==1 %if solve for tx power
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    Pt=-(-EbNo-rddb-marg-Ll+2*gain+fsl-La+228.6-snt); %solve link eq for Pt in dB
    PtW=10^(Pt/10); %convert Pt to Watts
    set(show_open,'String',PtW); %display watts
elseif (get(sel_ll,'Value'))==1 %if solve for Ll
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    Ll=Pt-EbNo-rddb-marg+2*gain+fsl-La+228.6-snt; %solve link eq for Ll in dB
    set(show_open,'String',Ll); %display
elseif (get(sel_eta,'Value'))==1 %if solve for eta
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    gt=-(Pt-EbNo-rddb-marg-Ll+fsl-La+228.6-snt); %solve link eq for gain term
    eta=((lamb^2)*(10^(gt/20)))/(4*(pi^2)*(rad^2)); %solve for eta (dim-less)
```

```
    set(show_open,'String',eta); %display
  elseif (get(sel_point,'Value'))==1 %if solve for La
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    La=Pt-EbNo-rddb-marg-Ll+2*gain+fsl+228.6-snt; %solve link eq for La in dB
    set(show_open,'String',La); %display
  elseif (get(sel_rad,'Value'))==1 %if solve for radius
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    gt=-(Pt-EbNo-rddb-marg-Ll+fsl-La+228.6-snt); %solve link eq for gain term
    rad=sqrt(((lamb^2)*(10^(gt/20)))/(4*(pi^2)*eta)); %solve for radius in m
    set(show_open,'String',rad); %display
  elseif (get(sel_snt,'Value'))==1 %if solve for snt
    gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2)); %compute gain
    fsl=-20*log10(4*pi*range/lamb); %solve free space loss NOTE NEGATIVE SIGN!
    snt=Pt-EbNo-rddb-marg-Ll+2*gain+fsl-La+228.6; %solve link eq for SNT in dB-K
    set(show_open,'String',snt); %display
end %determine which button pushed and solve for it


%Other computations of interest (now that we have all values!)
%over-rides above computations
%compute free space loss
fsl=-20*log10(4*pi*range/lamb); %NOTE NEGATIVE SIGN!
%compute antenna gain (one antenna!)
gain=10*log10((4*(pi^2)*(rad^2)*eta)/(lamb^2));
%compute 3dB beamwidth (70*lambda/diameter)
beamwidth=35*lamb/rad;
%compute G/T figure of merit
gt=gain-snt; %dB units subtract!
%Display fsl, gain, 3dB beamwidth, G/T
set(show_fsl,'String',fsl);
set(show_antgain,'String',gain);
set(show_3dbbw,'String',beamwidth);
set(show_gtfom,'String',gt);


%Error checking (red text) on invalid open parameter values
%('main_calculate' case resets compute box to black at beginning of each computation)
if (EbNo<-1.59) & (get(sel_ebno,'Value')==1) %if EbNo invalid & open parameter
  set(show_open,fgc,[1 0 0]); %red in open parameter (the compute box)
elseif (Ll<0) & (get(sel_ll,'Value')==1)
  set(show_open,fgc,[1 0 0]);
elseif (La<0) & (get(sel_point,'Value')==1)
  set(show_open,fgc,[1 0 0]);
elseif ((eta<=0) | (eta>1)) & (get(sel_eta,'Value')==1) %hi or lo eta
  set(show_open,fgc,[1 0 0]);
elseif (marg<0) & (get(sel_m,'Value')==1)
  set(show_open,fgc,[1 0 0]);
```

```
    elseif (rad<0) & (get(sel_rad,'Value')==1)
       set(show_open,fgc,[1 0 0]);
    end %invalid value error check


  case 'graph_it', %action on clicking graph button
    %Re-calculate with current values
    %main_calculate gets current values of all parameters
    comm_sel_code3 main_calculate


    %Replace 'solved-for' parameter with show_open value (determine show_parameter)
    %this over-rides values found in main_calculate
    if (get(show_parameter,'String'))=='Range '
      range=1000*eval(get(show_open,'String')); %convert displayed range km to m
    elseif (get(show_parameter,'String'))=='Eb/No '
      EbNo=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))==' Rd '
      rd=1e6*eval(get(show_open,'String')); %convert displayed rd Mbps to bps
      rddb=10*log10(rd); %rddb in db-bps
    elseif (get(show_parameter,'String'))=='Margin'
      marg=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))=='Lambda'
      lamb=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))==' Pt '
      PtW=eval(get(show_open,'String')); %Pt displays watts
      Pt=10*log10(PtW); %convert Pt watts to dB
    elseif (get(show_parameter,'String'))==' Ll '
      Ll=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))==' eta '
      eta=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))==' SNT '
      snt=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))==' La '
      La=eval(get(show_open,'String'));
    elseif (get(show_parameter,'String'))=='Radius'
      rad=eval(get(show_open,'String'));
    end %determine 'solved-for' parameter


    %Open new gui
    comm_graph_gui

end %end switch
```

Appendix D: RF Graphs GUI

comm_graph_code.m

```
function comm_graph_code(action)
% COMM_GRAPH_CODE.M   This file supports the pushbutton operations of GUI
% named comm_graph_gui.m  Enter "help comm_graph_gui" to get more information.
%
% COMM_GRAPH_CODE.M is not a stand-alone m-file.
%
% Capt. Andy Feltman
% V6.3; Last Update: 27 Feb 99 (Matlab V5.2 PC)


%Very important!!  Declare global variables before switches
%the following globals imported from comm_sel_gui3
global range EbNo rd rddb marg lamb Pt PtW Ll eta snt La rad fsl gain beamwidth gt
%********************


%Do all findobj commands once! (gets the handles!)
show_range=findobj(gcbf,'Tag','show_range');
show_ebno=findobj(gcbf,'Tag','show_ebno');
show_rd=findobj(gcbf,'Tag','show_rd');
show_m=findobj(gcbf,'Tag','show_m');
show_wave=findobj(gcbf,'Tag','show_wave');
sel_wave=findobj(gcbf,'Tag','sel_wave');
show_pt=findobj(gcbf,'Tag','show_pt');
show_ll=findobj(gcbf,'Tag','show_ll');
show_eta=findobj(gcbf,'Tag','show_eta');
show_snt=findobj(gcbf,'Tag','show_snt');
show_point=findobj(gcbf,'Tag','show_point');
show_rad=findobj(gcbf,'Tag','show_rad');
show_fsl=findobj(gcbf,'Tag','show_fsl');
show_antgain=findobj(gcbf,'Tag','show_antgain');
show_3dbbw=findobj(gcbf,'Tag','show_3dbbw');
show_gtfom=findobj(gcbf,'Tag','show_gtfom');
push_to_load=findobj(gcbf,'Tag','push_to_load');
lin_l=findobj(gcbf,'Tag','lin_l');
log_x=findobj(gcbf,'Tag','log_x');
log_y=findobj(gcbf,'Tag','log_y');
log_l=findobj(gcbf,'Tag','log_l');
the_graph=findobj(gcbf,'Tag','the_graph');
y_select=findobj(gcbf,'Tag','y_select');
x_select=findobj(gcbf,'Tag','x_select');
low_lim=findobj(gcbf,'Tag','low_lim');
hi_lim=findobj(gcbf,'Tag','hi_lim');
```

```
switch(action)

case 'initial', %action on creation of GUI
    %called by CreateFcn callbacks of show_range and show_fsl,gain,3dbbw,gtfom
    %Shows imported parameter values
    set(show_range,'String',(range/1000)); %range is in meters
    set(show_ebno,'String',EbNo);
    set(show_rd,'String',(rd/1e6)); %data rate to absolute Mbps
    set(show_m,'String',marg);
    set(show_wave,'String',(.3/lamb)); %display freq in GHz
    set(sel_wave,'Value',1); %set dropdown box to freq
    set(show_pt,'String',PtW); %display watts
    set(show_ll,'String',Ll);
    set(show_eta,'String',eta);
    set(show_snt,'String',snt);
    set(show_point,'String',La);
    set(show_rad,'String',rad);
    set(show_fsl,'String',fsl);
    set(show_antgain,'String',gain);
    set(show_3dbbw,'String',beamwidth);
    set(show_gtfom,'String',gt);

case 'load_all', %action on hitting load button
    comm_graph_code initial %SAME CODE

case 'check_wave', %action on selecting freq or lambda drop-down box
    %This always recomputes: new=.3/old  even if same drop-down selected (bug!)
    if get(sel_wave,'Value')==1 %select frequency in GHz
        temp=.3/(eval(get(show_wave,'String'))); %don't 'save' freq as a variable
        set(show_wave,'String',temp);
    elseif get(sel_wave,'Value')==2 %select wavelength in meters
        lamb2=.3/(eval(get(show_wave,'String'))); %lamb2 separate from imported lamb
        %separate variable name (lamb2 vs lamb) allows re-load of imported values
        set(show_wave,'String',lamb2);
    end

case 'up_sys_param' %action on direct entry of reference parameter
    %recalculates system parameters (FSL, Gain, 3dB, G/T)
    %get reference parameter values needed to compute system parameters
    t_range=1000*eval(get(show_range,'String')); %convert range to m
    t_radius=eval(get(show_rad,'String')); %radius in m
    t_eta=eval(get(show_eta,'String')); %eta is dim-less
    t_snt=eval(get(show_snt,'String')); %snt is dB-K
    %Get lambda as wavelength in meters
    if get(sel_wave,'Value')==1 %if GHz displayed
        t_lamb=.3/(eval(get(show_wave,'String'))); %convert to m
```

91

```
    elseif get(sel_wave,'Value')==2 %if lambda displayed, just get it
      t_lamb=eval(get(show_wave,'String'));
    end
    %recalculate system parameters
    t_fsl=-20*log10(4*pi*t_range/t_lamb); %FSL; NOTE NEGATIVE SIGN!
    t_gain=10*log10((4*(pi^2)*(t_radius^2)*t_eta)/(t_lamb^2)); %Gain (1 antenna!)
    t_beamwidth=35*t_lamb/t_radius; %3dB beamwidth (70*lambda/diameter)
    t_gt=t_gain-t_snt; %G/T figure of merit; dB units subtract!
    %Display recalculated parameters
    set(show_fsl,'String',t_fsl);
    set(show_antgain,'String',t_gain);
    set(show_3dbbw,'String',t_beamwidth);
    set(show_gtfom,'String',t_gt);

  case 'change_limit', %action on entering new low or high x-axis value
    %Reserved for future; see 'plot_style' case

  case 'change_x', %action on x-axis parameter
    x_val=get(x_select,'Value'); %'Value' returns a double; don't need eval
    %Key to x_val:
    % 1--Range    2--Eb/No 3--Rd   4--Margin   5--Lambda 6--freq (GHz)
    % 7--Pt       8--Ll    9--eta 10--SNT      11--La
    %12--Radius   13--FSL    14--Gain 15--3dB BW   16--G/T

    %Recommend low and high limits based on selection
    %x_ranges is 16x2 matrix of low/hi limits (cols) for each x_val (row)
    x_ranges=[2000 10000;2 12;10 1000;.5 4.5;.001 .1;3 300;1 50;.5 3;.4 .8];
    x_ranges=[x_ranges;33 37;.5 3;.1 1;-230 -170;35 60;.1 1;5 50];

    t_lo=x_ranges(x_val,1); %extract suggested low limit
    t_hi=x_ranges(x_val,2); %extract suggested hi limit
    set(low_lim,'String',t_lo); %display
    set(hi_lim,'String',t_hi); %display

  case 'change_y', %action on y-axis parameter
    %Reserved for future; see 'plot_style' case

  case 'plot_style', %action on selecting plot style and executes plot routine
    set(gca,'NextPlot','Replace'); %'Replace' will clear current plot
    min_max_flags=[0 0 0 0]; %Flags and values for parameter valid limits
    %button check to determine selection of plot style
    tempuse=get(gcbo,'Tag'); %find which button was selected; 'gcbo' is self-reference
    if tempuse=='lin_l' %if linear selected
      set(gcbo,'Value',1); %1. turn on linear button
      set(log_x,'Value',0); %2. turn off log_x button
      set(log_y,'Value',0); %3. turn off log_y button
```

92

```
      set(log_l,'Value',0); %4. turn off log_log button
      plot_type=1; %variable for plotting if/then statements
   elseif tempuse=='log_x' %if log_x selected
      set(gcbo,'Value',1);
      set(lin_l,'Value',0);
      set(log_y,'Value',0);
      set(log_l,'Value',0);
      plot_type=2;
   elseif tempuse=='log_y' %if log_y selected
      set(gcbo,'Value',1);
      set(lin_l,'Value',0);
      set(log_x,'Value',0);
      set(log_l,'Value',0);
      plot_type=3;
   elseif tempuse=='log_l' %if log_log selected
      set(gcbo,'Value',1);
      set(lin_l,'Value',0);
      set(log_x,'Value',0);
      set(log_y,'Value',0);
      plot_type=4;
   end


%DO THE PLOT; need parameters, axis selections, x-limits, and plot it
%************
%get system parameters--separate names xxx2 doesn't overwrite globals imported
%from comm_sel_gui3
range2=1000*eval(get(show_range,'String')); %convert range to m
EbNo2=eval(get(show_ebno,'String')); %EbNo in dB
rd2=1e6*eval(get(show_rd,'String')); %convert rd to bits/sec
rddb2=10*log10(rd2); %rddb in db-bps
marg2=eval(get(show_m,'String')); %Margin in dB
%Get lambda as wavelength in meters
if get(sel_wave,'Value')==1 %if GHz displayed
   lamb2=.3/(eval(get(show_wave,'String'))); %convert to m
elseif get(sel_wave,'Value')==2 %if lambda displayed, just get it
   lamb2=eval(get(show_wave,'String'));
end
Pt2=10*log10(eval(get(show_pt,'String'))); %convert displayed watts to dB-W
Ll2=eval(get(show_ll,'String')); %Ll in dB
eta2=eval(get(show_eta,'String')); %eta is dim-less
snt2=eval(get(show_snt,'String')); %snt in dB-K
La2=eval(get(show_point,'String')); %La in dB
rad2=eval(get(show_rad,'String')); %antenna radius in m
fsl2=eval(get(show_fsl,'String')); %FSL in dB
gain2=eval(get(show_antgain,'String')); %gain in dB
beamwidth2=eval(get(show_3dbbw,'String')); %bw in deg
```

93

```matlab
gt2=eval(get(show_gtfom,'String')); %G/T in dB

%get x parameter and limits
x_val=get(x_select,'Value'); %axis select value
lo_x=eval(get(low_lim,'String')); %low limit
hi_x=eval(get(hi_lim,'String')); %hi limit
if lo_x>hi_x %low limit must be less than hi limit
   temp_lo_hi=[0 1;1 0]*[lo_x;hi_x]; %switch low and hi limits
   lo_x=temp_lo_hi(1); %re-assign and display low value
   set(low_lim,'String',lo_x);
   hi_x=temp_lo_hi(2); %re-assign and display hi value
   set(hi_lim,'String',hi_x);
end
%x_range is 21-element vector evenly linearly spaced between lo_x & hi_x values
%x_range will be passed into calculations to get 21-ele vect of y values
x_range=[lo_x:((hi_x-lo_x)/20):hi_x];
%get y parameter
y_val=get(y_select,'Value'); %axis select value

%MAJOR COMPUTATIONS HERE
if x_val==y_val %trivial cases (same x and y axes)
   y_range=x_range; %y_range is 21-ele vector
   %
   %Range (y=1) as fn of other parameters
elseif (y_val==1)
  if (y_val==1) & (x_val==2) %range as fn of Eb/No
    stuff=Pt2-x_range-rddb2-marg2-Ll2+2*gain2-La2+228.6-snt2;
    y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000; %1000 to convert m to km
  elseif (y_val==1) & (x_val==3) %range as fn of Rd (convert Mbps to dB-bps)
    stuff=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2+2*gain2-La2+228.6-snt2;
    y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
  elseif (y_val==1) & (x_val==4) %range as fn of M
    stuff=Pt2-EbNo2-rddb2-x_range-Ll2+2*gain2-La2+228.6-snt2;
    y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
  elseif (y_val==1) & (x_val==5) %range as fn of lambda
    temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta)./(x_range.^2));
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain-La2+228.6-snt2;
    y_range=((x_range/(4*pi)).*10.^(stuff/20))/1000;
  elseif (y_val==1) & (x_val==6) %range as fn of freq (almost same as lambda)
    temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta)./((.3./x_range).^2));
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain-La2+228.6-snt2;
    y_range=(((.3./x_range)/(4*pi)).*10.^(stuff/20))/1000;
  elseif (y_val==1) & (x_val==7) %range as fn of Pt (convert W to dB-W)
    stuff=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2+2*gain2-La2+228.6-snt2;
    y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
  elseif (y_val==1) & (x_val==8) %range as fn of Ll
```

```
        stuff=Pt2-EbNo2-rddb2-marg2-x_range+2*gain2-La2+228.6-snt2;
        y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
      elseif (y_val==1) & (x_val==9) %range as fn of eta
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain-La2+228.6-snt2;
        y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
      elseif (y_val==1) & (x_val==10) %range as fn of snt
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2-La2+228.6-x_range;
        y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
      elseif (y_val==1) & (x_val==11) %range as fn of La
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2-x_range+228.6-snt2;
        y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
      elseif (y_val==1) & (x_val==12) %range as fn of rad
        temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta)./(lamb2^2));
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain-La2+228.6-snt2;
        y_range=((lamb2/(4*pi))*10.^(stuff/20))/1000;
      end %y=1
    %
    %Eb/No (y=2) as fn of other parameters
    elseif (y_val==2)
      %set Min flag and value to Shannon limit
      min_max_flags=[1 -1.59 0 0];
      if (y_val==2) & (x_val==1) %EbNo as fn of range
        temp_fsl=-20*log10(4000*pi*x_range/lamb2);
        y_range=Pt2-rddb2-marg2-Ll2+2*gain2+temp_fsl-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==3) %EbNo as fn of Rd
        y_range=Pt2-10*log10(1e6*x_range)-marg2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==4) %EbNo as fn of marg
        y_range=Pt2-rddb2-x_range-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==5) %EbNo as fn of lamb
        temp_fsl=-20*log10(4*pi*range2./x_range);
        temp_gain=20*log10(4*(pi^2)*(rad2^2)*eta2./(x_range.^2));
        y_range=Pt2-rddb2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==6) %EbNo as fn of freq
        temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
        temp_gain=20*log10(4*(pi^2)*(rad2^2)*eta2./((.3./x_range).^2));
        y_range=Pt2-rddb2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==7) %EbNo as fn of Pt
        y_range=10*log10(x_range)-rddb2-marg2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==8) %EbNo as fn of Ll
        y_range=Pt2-rddb2-marg2-x_range+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==9) %EbNo as fn of eta
        temp_gain=20*log10(4*(pi^2)*(rad2^2)*x_range/(lamb2^2));
        y_range=Pt2-rddb2-marg2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
      elseif (y_val==2) & (x_val==10) %EbNo as fn of snt
        y_range=Pt2-rddb2-marg2-Ll2+2*gain2+fsl2-La2+228.6-x_range;
```

```
    elseif (y_val==2) & (x_val==11) %EbNo as fn of La
      y_range=Pt2-rddb2-marg2-Ll2+2*gain2+fsl2-x_range+228.6-snt2;
    elseif (y_val==2) & (x_val==12) %EbNo as fn of radius
      temp_gain=20*log10(4*(pi^2)*(x_range.^2)*eta2/(lamb2^2));
      y_range=Pt2-rddb2-marg2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
    end %y=2
  %
  %Rd (y=3) as fn of other parameters
elseif (y_val==3)
  if (y_val==3) & (x_val==1) %Rd as fn of range
    temp_fsl=-20*log10(4000*pi*x_range/lamb2);
    stuff=Pt2-EbNo2-marg2-Ll2+2*gain2+temp_fsl-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==2) %Rd as fn of EbNo
    stuff=Pt2-x_range-marg2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==4) %Rd as fn of marg
    stuff=Pt2-EbNo2-x_range-Ll2+2*gain2+fsl2-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==5) %Rd as fn of lamb
    temp_fsl=-20*log10(4*pi*range2./x_range);
    temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
    stuff=Pt2-EbNo2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==6) %Rd as fn of freq
    temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
    temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
    stuff=Pt2-EbNo2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==7) %Rd as fn of Pt
    stuff=10*log10(x_range)-EbNo2-marg2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==8) %Rd as fn of Ll
    stuff=Pt2-EbNo2-marg2-x_range+2*gain2+fsl2-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==9) %Rd as fn of eta
    temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
    stuff=Pt2-EbNo2-marg2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==10) %Rd as fn of snt
    stuff=Pt2-EbNo2-marg2-Ll2+2*gain2+fsl2-La2+228.6-x_range;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==11) %Rd as fn of La
    stuff=Pt2-EbNo2-marg2-Ll2+2*gain2+fsl2-x_range+228.6-snt2;
    y_range=(10.^(stuff/10))/1e6;
  elseif (y_val==3) & (x_val==12) %Rd as fn of radius
```

```
        temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
        stuff=Pt2-EbNo2-marg2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
        y_range=(10.^(stuff/10))/1e6;
      end %y=3
      %
      %Margin (y=4) as fn of other parameters
    elseif (y_val==4)
      %set Min flag and value to zero
      min_max_flags=[1 0 0 0];
      if (y_val==4) & (x_val==1) %Margin as fn of range
        temp_fsl=-20*log10(4000*pi*x_range/lamb2);
        y_range=Pt2-EbNo2-rddb2-Ll2+2*gain2+temp_fsl-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==2) %Margin as fn of EbNo
        y_range=Pt2-x_range-rddb2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==3) %Margin as fn of Rd
        y_range=Pt2-EbNo2-10*log10(1e6*x_range)-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==5) %Margin as fn of lamb
        temp_fsl=-20*log10(4*pi*range2./x_range);
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
        y_range=Pt2-EbNo2-rddb2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==6) %Margin as fn of freq
        temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
        y_range=Pt2-EbNo2-rddb2-Ll2+temp_gain+temp_fsl-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==7) %Margin as fn of Pt
        y_range=10*log10(x_range)-EbNo2-rddb2-Ll2+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==8) %Margin as fn of Ll
        y_range=Pt2-EbNo2-rddb2-x_range+2*gain2+fsl2-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==9) %Margin as fn of eta
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
        y_range=Pt2-EbNo2-rddb2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
      elseif (y_val==4) & (x_val==10) %Margin as fn of snt
        y_range=Pt2-EbNo2-rddb2-Ll2+2*gain2+fsl2-La2+228.6-x_range;
      elseif (y_val==4) & (x_val==11) %Margin as fn of La
        y_range=Pt2-EbNo2-rddb2-Ll2+2*gain2+fsl2-x_range+228.6-snt2;
      elseif (y_val==4) & (x_val==12) %Margin as fn of radius
        temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
        y_range=Pt2-EbNo2-rddb2-Ll2+temp_gain+fsl2-La2+228.6-snt2;
      end %y=4
      %
      %Lambda (y=5) as fn of other parameters
    elseif (y_val==5)
      if (y_val==5) & (x_val==1) %Lambda as fn of range
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
        y_range=10.^((1/20)*(stuff-
20*log10(4000*pi*x_range)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
```

```
    elseif (y_val==5) & (x_val==2) %Lambda as fn of EbNo
       stuff=Pt2-x_range-rddb2-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==3) %Lambda as fn of Rd
       stuff=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==4) %Lambda as fn of marg
       stuff=Pt2-EbNo2-rddb2-x_range-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==6) %Lambda as fn of freq (trivial)
       y_range=.3./x_range;
     elseif (y_val==5) & (x_val==7) %Lambda as fn of Pt
       stuff=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==8) %Lambda as fn of Ll
       stuff=Pt2-EbNo2-rddb2-marg2-x_range-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==9) %Lambda as fn of eta
       stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*x_range)));
     elseif (y_val==5) & (x_val==10) %Lambda as fn of snt
       stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-x_range;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==11) %Lambda as fn of La
       stuff=Pt2-EbNo2-rddb2-marg2-Ll2-x_range+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
     elseif (y_val==5) & (x_val==12) %Lambda as fn of rad
       stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(x_range.^2)*eta2)));
     end %y=5
     %
     %Freq (y=6) as fn of other parameters--similar to lambda (y=5) code
   elseif (y_val==6)
     if (y_val==6) & (x_val==1) %Freq as fn of range
       stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
       y_range=10.^((1/20)*(stuff-
20*log10(4000*pi*x_range)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
```

98

```
    y_range=.3./y_range; %convert to freq in GHz
  elseif (y_val==6) & (x_val==2) %Freq as fn of EbNo
    stuff=Pt2-x_range-rddb2-marg2-Ll2-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==3) %Freq as fn of Rd
    stuff=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==4) %Freq as fn of marg
    stuff=Pt2-EbNo2-rddb2-x_range-Ll2-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==5) %Freq as fn of lambda (trivial)
    y_range=.3./x_range;
  elseif (y_val==6) & (x_val==7) %Freq as fn of Pt
    stuff=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==8) %Freq as fn of Ll
    stuff=Pt2-EbNo2-rddb2-marg2-x_range-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==9) %Freq as fn of eta
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*x_range)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==10) %Freq as fn of snt
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-x_range;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==11) %Freq as fn of La
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2-x_range+228.6-snt2;
    y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(rad2^2)*eta2)));
    y_range=.3./y_range;
  elseif (y_val==6) & (x_val==12) %Freq as fn of rad
    stuff=Pt2-EbNo2-rddb2-marg2-Ll2-La2+228.6-snt2;
```

```
        y_range=10.^((1/20)*(stuff-
20*log10(4*pi*range2)+20*log10(4*(pi^2)*(x_range.^2)*eta2)));
        y_range=.3./y_range;
      end %y=6
      %
      %Pt (y=7) as fn of other parameters
    elseif (y_val==7)
      if (y_val==7) & (x_val==1) %Pt as fn of range
        temp_fsl=-20*log10(4000*pi*x_range/lamb2);
        stuff=EbNo2+rddb2+marg2+Ll2-2*gain2-temp_fsl+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==2) %Pt as fn of EbNo
        stuff=x_range+rddb2+marg2+Ll2-2*gain2-fsl2+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==3) %Pt as fn of Rd
        stuff=EbNo2+10*log10(1e6*x_range)+marg2+Ll2-2*gain2-fsl2+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==4) %Pt as fn of marg
        stuff=EbNo2+rddb2+x_range+Ll2-2*gain2-fsl2+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==5) %Pt as fn of lambda
        temp_fsl=-20*log10(4*pi*range2./x_range);
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
        stuff=EbNo2+rddb2+marg2+Ll2-temp_gain-temp_fsl+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==6) %Pt as fn of freq
        temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
        stuff=EbNo2+rddb2+marg2+Ll2-temp_gain-temp_fsl+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==8) %Pt as fn of Ll
        stuff=EbNo2+rddb2+marg2+x_range-2*gain2-fsl2+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==9) %Pt as fn of eta
        temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
        stuff=EbNo2+rddb2+marg2+Ll2-temp_gain-fsl2+La2-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==10) %Pt as fn of snt
        stuff=EbNo2+rddb2+marg2+Ll2-2*gain2-fsl2+La2-228.6+x_range;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==11) %Pt as fn of La
        stuff=EbNo2+rddb2+marg2+Ll2-2*gain2-fsl2+x_range-228.6+snt2;
        y_range=10.^(stuff/10);
      elseif (y_val==7) & (x_val==12) %Pt as fn of radius
        temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)./(lamb2^2));
        stuff=EbNo2+rddb2+marg2+Ll2-temp_gain-fsl2+La2-228.6+snt2;
```

```matlab
      y_range=10.^(stuff/10);
    end %y=7
    %
    %Ll (y=8) as fn of other parameters
elseif (y_val==8)
    %set Min flag and value to zero
    min_max_flags=[1 0 0 0];
    if (y_val==8) & (x_val==1) %Ll as fn of range
      temp_fsl=-20*log10(4000*pi*x_range/lamb2);
      y_range=Pt2-EbNo2-rddb2-marg2+2*gain2+temp_fsl-La2+228.6-snt;
    elseif (y_val==8) & (x_val==2) %Ll as fn of EbNo
      y_range=Pt2-x_range-rddb2-marg2+2*gain2+fsl2-La2+228.6-snt;
    elseif (y_val==8) & (x_val==3) %Ll as fn of Rd
      y_range=Pt2-EbNo2-10*log10(1e6*x_range)-marg2+2*gain2+fsl2-La2+228.6-snt;
    elseif (y_val==8) & (x_val==4) %Ll as fn of marg
      y_range=Pt2-EbNo2-rddb2-x_range+2*gain2+fsl2-La2+228.6-snt;
    elseif (y_val==8) & (x_val==5) %Ll as fn of lambda
      temp_fsl=-20*log10(4*pi*range2./x_range);
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
      y_range=Pt2-EbNo2-rddb2-marg2+temp_gain+temp_fsl-La2+228.6-snt;
    elseif (y_val==8) & (x_val==6) %Ll as fn of freq
      temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
      y_range=Pt2-EbNo2-rddb2-marg2+temp_gain+temp_fsl-La2+228.6-snt;
    elseif (y_val==8) & (x_val==7) %Ll as fn of Pt
      y_range=10*log10(x_range)-EbNo2-rddb2-marg2+2*gain2+fsl2-La2+228.6-snt;
    elseif (y_val==8) & (x_val==9) %Ll as fn of eta
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2+temp_gain+fsl2-La2+228.6-snt;
    elseif (y_val==8) & (x_val==10) %Ll as fn of snt
      y_range=Pt2-EbNo2-rddb2-marg2+2*gain2+fsl2-La2+228.6-x_range;
    elseif (y_val==8) & (x_val==11) %Ll as fn of La
      y_range=Pt2-EbNo2-rddb2-marg2+2*gain2+fsl2-x_range+228.6-snt2;
    elseif (y_val==8) & (x_val==12) %Ll as fn of radius
      temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2+temp_gain+fsl2-La2+228.6-snt;
    end %y=8
    %
    %eta (y=9) as fn of other parameters
elseif (y_val==9)
    %set Min and Max flags and value to zero to one
    min_max_flags=[1 0 1 1];
    if (y_val==9) & (x_val==1) %eta as fn of range
      temp_fsl=-20*log10(4000*pi*x_range/lamb2);
      stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
      y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
```

```matlab
      elseif (y_val==9) & (x_val==2) %eta as fn of Eb/No
         stuff=Pt2-x_range-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==3) %eta as fn of Rd
         stuff=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);  .
      elseif (y_val==9) & (x_val==4) %eta as fn of marg
         stuff=Pt2-EbNo2-rddb2-x_range-Ll2+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==5) %eta as fn of lambda
         temp_fsl=-20*log10(4*pi*range2./x_range);
         stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
         y_range=((x_range.^2)/(4*(pi^2)*(rad2^2))).*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==6) %eta as fn of freq
         temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
         stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
         y_range=(((.3./x_range).^2)/(4*(pi^2)*(rad2^2))).*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==7) %eta as fn of Pt
         stuff=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==8) %eta as fn of Ll
         stuff=Pt2-EbNo2-rddb2-marg2-x_range+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==10) %eta as fn of snt
         stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-x_range;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==11) %eta as fn of La
         stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-x_range+228.6-snt2;
         y_range=((lamb2^2)/(4*(pi^2)*(rad2^2)))*10.^(-stuff/20);
      elseif (y_val==9) & (x_val==12) %eta as fn of radius
         stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
         y_range=((lamb2^2)./(4*(pi^2)*(x_range.^2)))*10.^(-stuff/20);
      end %y=9
   %
   %snt (y=10) as fn of other parameters
   elseif (y_val==10)
      if (y_val==10) & (x_val==1) %snt as fn of range
         temp_fsl=-20*log10(4000*pi*x_range/lamb2);
         y_range=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2+temp_fsl-La2+228.6;
      elseif (y_val==10) & (x_val==2) %snt as fn of EbNo
         y_range=Pt2-x_range-rddb2-marg2-Ll2+2*gain2+fsl2-La2+228.6;
      elseif (y_val==10) & (x_val==3) %snt as fn of Rd
         y_range=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2+2*gain2+fsl2-La2+228.6;
      elseif (y_val==10) & (x_val==4) %snt as fn of marg
         y_range=Pt2-EbNo2-rddb2-x_range-Ll2+2*gain2+fsl2-La2+228.6;
      elseif (y_val==10) & (x_val==5) %snt as fn of lambda
```

```
      temp_fsl=-20*log10(4*pi*range2./x_range);
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6;
   elseif (y_val==10) & (x_val==6) %snt as fn of freq
      temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+temp_fsl-La2+228.6;
   elseif (y_val==10) & (x_val==7) %snt as fn of Pt
      y_range=x_range-EbNo2-rddb2-marg2-Ll2+2*gain2+fsl2-La2+228.6;
   elseif (y_val==10) & (x_val==8) %snt as fn of Ll
      y_range=Pt2-EbNo2-rddb2-marg2-x_range+2*gain2+fsl2-La2+228.6;
   elseif (y_val==10) & (x_val==9) %snt as fn of eta
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+fsl2-La2+228.6;
   elseif (y_val==10) & (x_val==11) %snt as fn of La
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2+fsl2-x_range+228.6;
   elseif (y_val==10) & (x_val==12) %snt as fn of radius
      temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+fsl2-La2+228.6;
   end %y=10
   %
   %La (y=11) as fn of other parameters
 elseif (y_val==11)
   %set Min flag and value to zero
   min_max_flags=[1 0 0 0];
   if (y_val==11) & (x_val==1) %La as fn of range
      temp_fsl=-20*log10(4000*pi*x_range/lamb2);
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2+temp_fsl+228.6-snt2;
    elseif (y_val==11) & (x_val==2) %La as fn of EbNo
      y_range=Pt2-x_range-rddb2-marg2-Ll2+2*gain2+fsl2+228.6-snt2;
    elseif (y_val==11) & (x_val==3) %La as fn of Rd
      y_range=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2+2*gain2+fsl2+228.6-
snt2;
    elseif (y_val==11) & (x_val==4) %La as fn of marg
      y_range=Pt2-EbNo2-rddb2-x_range-Ll2+2*gain2+fsl2+228.6-snt2;
    elseif (y_val==11) & (x_val==5) %La as fn of lambda
      temp_fsl=-20*log10(4*pi*range2./x_range);
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+temp_fsl+228.6-snt2;
    elseif (y_val==11) & (x_val==6) %La as fn of freq
      temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+temp_fsl+228.6-snt2;
    elseif (y_val==11) & (x_val==7) %La as fn of Pt
      y_range=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2+2*gain2+fsl2+228.6-snt2;
    elseif (y_val==11) & (x_val==8) %La as fn of Ll
```

```
      y_range=Pt2-EbNo2-rddb2-marg2-x_range+2*gain2+fsl2+228.6-snt2;
   elseif (y_val==11) & (x_val==9) %La as fn of eta
      temp_gain=20*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+fsl2+228.6-snt2;
   elseif (y_val==11) & (x_val==10) %La as fn of snt
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+2*gain2+fsl2+228.6-x_range;
   elseif (y_val==11) & (x_val==12) %La as fn of radius
      temp_gain=20*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
      y_range=Pt2-EbNo2-rddb2-marg2-Ll2+temp_gain+fsl2+228.6-snt2;
   end %y=11
%
%radius (y=12) as fn of other parameters
elseif (y_val==12)
   %set Min flag and value to zero
   min_max_flags=[1 0 0 0];
   if (y_val==12) & (x_val==1) %radius as fn of range
      temp_fsl=-20*log10(4000*pi*x_range/lamb2);
      stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==2) %radius as fn of EbNo
      stuff=Pt2-x_range-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==3) %radius as fn of Rd
      stuff=Pt2-EbNo2-10*log10(1e6*x_range)-marg2-Ll2+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==4) %radius as fn of marg
      stuff=Pt2-EbNo2-rddb2-x_range-Ll2+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==5) %radius as fn of lambda
      temp_fsl=-20*log10(4*pi*range2./x_range);
      stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
      y_range=sqrt(((x_range.^2)/(4*(pi^2)*eta2)).*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==6) %radius as fn of freq
      temp_fsl=-20*log10(4*pi*range2./(.3./x_range));
      stuff=Pt2-EbNo2-rddb2-marg2-Ll2+temp_fsl-La2+228.6-snt2;
      y_range=sqrt((((.3./x_range).^2)/(4*(pi^2)*eta2)).*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==7) %radius as fn of Pt
      stuff=10*log10(x_range)-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==8) %radius as fn of Ll
      stuff=Pt2-EbNo2-rddb2-marg2-x_range+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==9) %radius as fn of eta
      stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-snt2;
      y_range=sqrt(((lamb2^2)./(4*(pi^2)*x_range))*10.^(-stuff/20));
   elseif (y_val==12) & (x_val==10) %radius as fn of snt
```

```
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-La2+228.6-x_range;
        y_range=sqrt((((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
      elseif (y_val==12) & (x_val==11) %radius as fn of La
        stuff=Pt2-EbNo2-rddb2-marg2-Ll2+fsl2-x_range+228.6-snt2;
        y_range=sqrt((((lamb2^2)/(4*(pi^2)*eta2))*10.^(-stuff/20));
      end %y=12
      %
      %FSL (y=13) as classic mathematical function definition of other parameters
    elseif (y_val==13)
      if (y_val==13) & (x_val==1) %FSL as fn of range
        y_range=-20*log10(4000*pi*x_range/lamb2);
      elseif (y_val==13) & (x_val==5) %FSL as fn of lamb
        y_range=-20*log10(4*pi*range2./x_range);
      elseif (y_val==13) & (x_val==6) %FSL as fn of freq
        y_range=-20*log10(4*pi*range2./(.3./x_range));
      else y_range=fsl2*ones(1,21); %NFR--FSL is constant
      end %y=13
      %
      %Gain (y=14) as classic mathematical function definition of other parameters
    elseif (y_val==14)
      if (y_val==14) & (x_val==5) %gain as fn of lamb
        y_range=10*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2));
      elseif (y_val==14) & (x_val==6) %gain as fn of lamb
        y_range=10*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2));
      elseif (y_val==14) & (x_val==9) %gain as fn of eta
        y_range=10*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2));
      elseif (y_val==14) & (x_val==12) %gain as fn of radius
        y_range=10*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2));
      else y_range=gain2*ones(1,21); %NFR--gain is constant
      end %y=14
      %
      %3dB beamwidth (y=15) as classic mathematical function definition of other
parameters
    elseif (y_val==15)
      if (y_val==15) & (x_val==5) %3dB as fn of lamb
        y_range=35.*x_range./rad2;
      elseif (y_val==15) & (x_val==6) %3dB as fn of freq
        y_range=35.*(.3./x_range)./rad2;
      elseif (y_val==15) & (x_val==12) %3dB as fn of radius
        y_range=35.*lamb2./x_range;
      else y_range=beamwidth2*ones(1,21); %NFR--3dB is constant
      end %y=15
      %
      %G/T (y=16) as as classic mathematical function definition of other parameters
    elseif (y_val==16)
      if (y_val==16) & (x_val==5) %gain as fn of lamb
```

```matlab
      y_range=10*log10((4*(pi^2)*(rad2^2)*eta2)./(x_range.^2))-snt2;
   elseif (y_val==16) & (x_val==6) %gain as fn of freq
      y_range=10*log10((4*(pi^2)*(rad2^2)*eta2)./((.3./x_range).^2))-snt2;
   elseif (y_val==16) & (x_val==9) %gain as fn of eta
      y_range=10*log10((4*(pi^2)*(rad2^2)*x_range)/(lamb2^2))-snt2;
   elseif (y_val==16) & (x_val==10) %G/T as fn of snt
      y_range=gain2-x_range; %subtract because dB
   elseif (y_val==16) & (x_val==12) %gain as fn of radius
      y_range=10*log10((4*(pi^2)*(x_range.^2)*eta2)/(lamb2^2))-snt2;
   else y_range=gt2*ones(1,21); %NFR--G/T is constant
   end %y=16
   %
end %major computations


%THE ACTUAL PLOTTING
if plot_type==1 %Linear
   plot(x_range,y_range); %the actual plot; x_range and y_range are 1x21 vectors
   set(gca,'NextPlot','Add'); %'Add' allows overwrite of multiple graphs
   if min_max_flags(1)==1 %check for min
      plot(x_range,min_max_flags(2),'r^'); %plot triangles at min
   end
   if min_max_flags(3)==1 %check for max
      plot(x_range,min_max_flags(4),'rv'); %plot upsidedown triangles at max
   end
elseif plot_type==2 %Log X-axis
   semilogx(x_range,y_range);
   set(gca,'NextPlot','Add');
   if min_max_flags(1)==1
      semilogx(x_range,min_max_flags(2),'r^');
   end
   if min_max_flags(3)==1
      semilogx(x_range,min_max_flags(4),'rv');
   end
elseif plot_type==3 %Log Y-axis
   %Avoid error of trying to plot negative y-values on log scale
   poss=sign(y_range); %poss is 1x21 with elements = 1 if y_range>0, 0 if =0, -1 if <0
   if sum(poss)<21 %if not all elements > 0 (i.e., if there are zero/negative y values)
      %newx and newy are subsets of x_range and y_range that contain only the positive
      %elements of x_range and y_range (if an element of y_range is 0 or negative, do
      %not add the (x_range,y_range) pair to newx and newy)
      newx=[];newy=[]; %initialize vectors
      for i=1:21, %count thru y_range--known 21 elements
         if poss(i)==1 %if y_range element >= 0 (positive)
            newx=[newx x_range(i)]; %add x_range element to newx
            newy=[newy y_range(i)]; %add y_range element to newy
         end %if poss=1; nothing happens if y_range element is zero or negative
```

106

```
        end %for loop
        x_range=newx;y_range=newy; %rename newx/newy as x_range/y_range
      end %if not all elements of original y_range > 0
      semilogy(x_range,y_range); %the plot--all y_range elements are positive
      set(gca,'NextPlot','Add'); %allows overwrite on plot
      %KNOW ALL MIN VALUE LIMITS ARE ZERO OR NEG--DO NOT PLOT
      %(because negative values give warning on log plots)
      if min_max_flags(3)==1
        plot(x_range,min_max_flags(4),'rv');
      end
    elseif plot_type==4 %Log-Log
      poss=sign(y_range);
      if sum(poss)<21 %if not all elements > 0
        newx=[];newy=[]; %initialize 'positive' vectors
        for i=1:21, %count thru y_range--known 21 elements
          if poss(i)==1 %if y_range element >= 0 (positive)
            newx=[newx x_range(i)];
            newy=[newy y_range(i)];
          end %if poss=1
        end %for loop
        x_range=newx;y_range=newy; %redefine as 'positive' vectors
      end %if not all elements > 0
      loglog(x_range,y_range); %the plot--all y_range elements are positive
      set(gca,'NextPlot','Add');
      %again don't check for minimum values
      if min_max_flags(3)==1
        semilogx(x_range,min_max_flags(4),'rv');
      end
    end
    set(gca,'XLim',[lo_x hi_x]); %set x-axis limits to current values

end %end case
```

BIBLIOGRAPHY

[AdR87] Adams, W. S., and Rider, L. "Circular Polar Constellations Providing Continuous Single or Multiple Coverage Above a Specified Altitude," *The Journal of the Astronautical Sciences*, Vol. 35, #2 (Apr-Jun), pp. 155-192 (1987).

[Ara98] Araki, T. and others. "Latest Results and Trade-Off of High-Power Optical Fiber Amplifiers for Optical Interorbit Communications," in *Free-Space Laser Communication Technologies X*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 3266, pp. 42-48 (1998).

[Bai98] Baister, G. C. and others. "Optical Communications Terminals for Multimedia Applications," in *Free-Space Laser Communication Technologies X*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 3266, pp. 135-145 (1998).

[Bis98] Biswas, A. and others. "Results of the STRV-2 Lasercom Terminal Evaluation Tests," in *Free-Space Laser Communication Technologies X*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 3266, pp. 2-13 (1998).

[Cha98] Chan, E. Y. and others. "Application of COTS High-Power Laser Diodes and Driver for a Free-Space Laser Communication Terminal," in *Free-Space Laser Communication Technologies X*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 3266, pp. 54-69 (1998).

[DeK74] Degnan, John J., and Klein, Bernard J. "Optical Antenna Gain. 2: Receiving Antennas," *Applied Optics*, Vol. 13, #10 (October), pp. 2397-2401 (1974).

[Hug98] Hughes, Mark T., Col. "Discoverer II: Space-Based GMTI/SAR Demonstration Program." Slide Show. http://www.darpa.mil/tto/dis2-docs.htm and http://www.darpa.mil/tto/dis2/br3/sld001.htm (page dates: 24 Jun 98). Last access: 18 Feb 99.

[Joh93] Johnson, Richard C., ed. Antenna Engineering Handbook (Third Edition). New York: McGraw Hill, Inc., 1993.

[Kat87] Katzman, Morris, ed. Laser Satellite Communications. Englewood Cliffs NJ: Prentice-Hall, Inc., 1987.

[Kim98] Kim, I. I., and others. "Scintillation Measurements Performed During the Limited-Visibility Lasercom Experiment," in *Free-Space Laser Communication Technologies X*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 3266, pp. 209-220 (1998).

[KlD74] Klein, Bernard J., and Degnan, John J. "Optical Antenna Gain. 1: Transmitting Antennas," *Applied Optics*, Vol. 13, #9 (September), pp. 2134-2141 (1974).

[Kor97] Korevaar, Eric and others. "Description of STRV-2 Lasercom Flight Hardware," in *Free-Space Laser Communication Technologies IX*, G. Stephen Mecherle, Ed., Proceedings of SPIE Vol. 2990, pp. 38-49 (1997).

[MoO87] Morrison, David and Owen, Tobias. The Planetary System. Reading MA: Addison-Wesley, 1987, pp. 202-208.

[PeZ95] Peterson, Roger L., Ziemer, Rodger E., and Borth, David E. Introduction to Spread Spectrum Communications. Englewood Cliffs NJ: Prentice-Hall, Inc., 1995.

[Ric95] Richaria, M. Satellite Communication Systems: Design Principles. New York: McGraw Hill, Inc., 1995.

[Skl88] Sklar, Bernard. Digital Communications: Fundamentals and Applications. Englewood Cliffs, NJ: PTR Prentice Hall, 1988.

[Whe98] Whelan, David. "Discoverer II: Global Precision Surveillance." Industry Day Briefing Slide Show. http://www.darpa.mil/tto/dis2-docs.htm and http://www.darpa.mil/tto/dis2/br2/sld001.htm (page dates: 24 Jun 98). Last access: 18 Feb 99.

Vita

Captain Andrew J. Feltman was born in February 1968 in Evanston, Illinois. He graduated from George Mason Jr.-Sr. High School in Falls Church, Virginia in June 1986. He entered undergraduate studies at the Massachusetts Institute of Technology (MIT) in Cambridge, Massachusetts where he graduated with a Bachelor of Science degree in Electrical Engineering in June 1990. He was commissioned through the Detachment 365 AFROTC at MIT.

He attended Basic Communications-Computer Systems Officers Training (BCOT) school at Keesler AFB, MS from April to August 1991. In August 1991, he was assigned to the 1845th Engineering Installation Group, Tinker AFB, OK in the Distributions Systems Section. In July 1994, he was assigned to Air Force Flight Standards Agency, Andrews AFB, MD in the Air Traffic Control and Landing Systems Evaluation Division. In August 1997, he entered the Electrical Engineering program, School of Engineering, Air Force Institute of Technology. Upon graduation, he will be assigned as an instructor at BCOT, Keesler AFB, MS.

Permanent Address: 707 Hillwood Avenue
Falls Church VA 22042

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE TRADE-OFF ANALYSIS OF COMMUNICATIONS CAPABILITIES OF INTER-SATELLITE LINKS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Andrew J. Feltman, Captain, USAF			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P St. WPAFB OH 45433-7765			**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/99M-08
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Major Ronald Delap SMC/XRI 180 Skynet Way, Ste. 2234 Los Angeles AFB CA 90245-4687 ph 310/363-0731			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES Major Richard Raines richard.raines@aift.af.mil ph 937/255-3636 ext 4715			
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

This thesis designs, develops, and uses software Graphical User Interfaces (GUIs) to analyze the communications capabilities of Radio Frequency (RF) Inter-Satellite Links (ISLs). The GUIs are geared towards analyzing the proposed ISLs of the Discoverer II program, but are general enough to permit analysis of any free-space RF ISLs. Discoverer II is a demonstration program of low-earth orbiting satellites and is primarily focused on satellite-based sensor technology. This thesis shows RF ISLs can meet the program requirement to broadcast the sensor data back to CONUS in near-real-time. The GUIs operate in real-time and explore trade-offs in the communications capability by varying the requirements, parameters, and components of the system design. Three GUIs are developed. The first GUI uses parameters from an assumed satellite constellation geometry to calculate the operating range of the ISLs. The second GUI solves for an unknown parameter of the communications system when all other parameters are given. The third GUI extends the analysis capability by plotting the trade-off between two parameters over a specified range of data. The analysis of the Discoverer II ISLs indicates that antenna radius is a very significant factor in the trade-off analysis, as well as being an important satellite design parameter.

14. SUBJECT TERMS Inter-satellite links, link equation, link equation trade-offs, radio frequency communications systems			15. NUMBER OF PAGES 121
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL